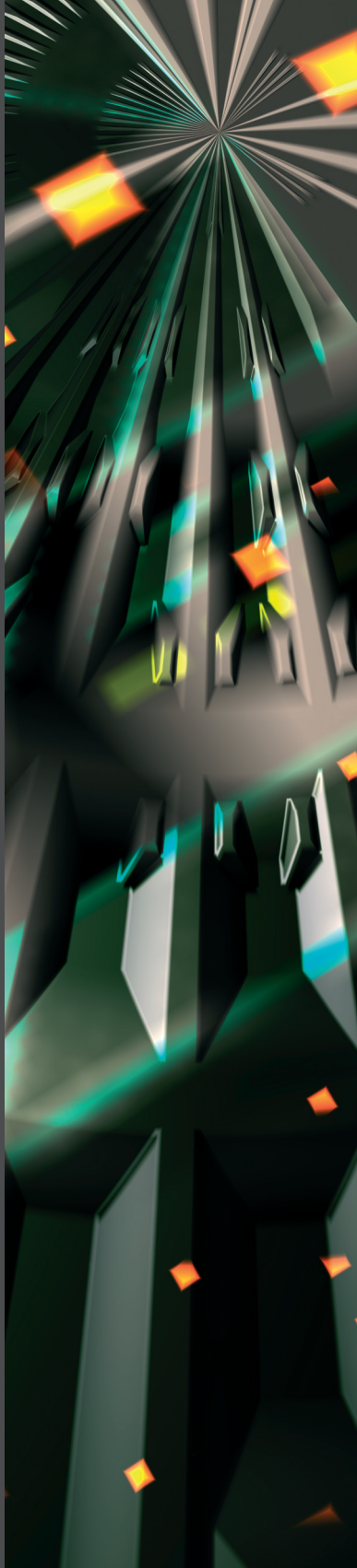

Christian Piguet
Heinz Hügli
DU ZERO
à l'ordinateur
Une brève histoire du calcul
Presses polytechniques et universitaires romandes

Ce livre expose de manière claire et progressive la façon dont sont nées les idées qui ont conduit à la réalisation des ordinateurs et des microprocesseurs actuels. Il offre au lecteur les clés lui permettant de mieux comprendre la nature et le fonctionnement de ces machines.
L'histoire du calcul débute avec l'apparition des systèmes de numération et l'invention du zéro, elle se poursuit avec la conception des premières machines à calculer mécaniques et électro-mécaniques, puis aboutit enfin aux machines électroniques. Ces dernières apparaissent d'abord sous la forme d'ordinateurs primitifs constitués de volumineux tubes à vides, avant d'évoluer vers les machines sophistiquées que nous connaissons aujourd'hui, composées de millions de minuscules transistors en silicium.
Richement illustrée, proposant fils conducteurs et détails de fonctionnement, l'histoire exposée dans cet ouvrage intéressera tous ceux et celles curieux de connaître la genèse et l'avènement des ordinateurs.

Christian Piguet est chef du secteur « Ultra Low Power » au Centre Suisse d'Electronique et de Microtechnique S.A. (CSEM) et professeur à l'Ecole polytechnique fédérale de Lausanne.

Heinz Hügli est directeur de recherche et professeur associé à l'Institut de microtechnique (IMT) de l'Université de Neuchâtel où il dirige le laboratoire de Reconnaissance des formes.

Presses polytechniques et universitaires romandes



DU ZERO
à l'ordinateur

Christian Piguet Heinz Hügli

DU ZERO à l'ordinateur

Une brève histoire du calcul

Presses polytechniques et universitaires romandes

Les auteurs et l'éditeur remercient l'Ecole polytechnique fédérale de Lausanne et l'Université de Neuchâtel qui ont soutenu la publication de cet ouvrage

Egalement parus aux Presses polytechniques et universitaires romandes:

Compter avec des cailloux
Alain Schärlig

Compter avec des jetons
Alain Schärlig

Lettres à une princesse d'Allemagne
Leonhard Euler

Un traité médiéval sur les carrés magiques
Jacques Sesiano

Les Presses polytechniques et universitaires romandes sont une fondation scientifique dont le but est principalement la diffusion des travaux de l'Ecole polytechnique fédérale de Lausanne ainsi que d'autres universités et écoles d'ingénieurs francophones.
Le catalogue de leurs publications peut être obtenu par courrier aux Presses polytechniques et universitaires romandes: EPFL – Centre Midi, CH-1015 Lausanne, par E-Mail à ppur@epfl.ch, par téléphone au (0)21 693 41 40, ou par fax au (0)21 693 40 27.

www.ppur.org

Première édition
ISBN 2-88074-469-5

Imprimé en Suisse

PRÉFACE

Le calcul a été et continue d'être une activité principale de l'*homo sapiens*. Depuis la nuit des temps, l'homme a compté ses proies dans la forêt, et plus tard ses bœufs dans sa ferme. Le commerce, une activité fondamentale pour le développement des peuples, est basé sur le calcul de profits et de pertes. Les civilisations qui ont maîtrisé le calcul, et l'ont appliqué à la défense et aux conquêtes, ont joué un rôle dominateur pour de longues périodes dans l'histoire.

La révolution industrielle du XVIII^{e} siècle trouve ses fondations dans le progrès des sciences appliquées et sur la mesure et le calcul des grandeurs. C'est le calcul de la pression de la vapeur qui permet de brider sa puissance pour faire tourner le volant et les métiers. De même que la révolution industrielle a changé l'économie du monde, la révolution informatique est en train de la bouleverser encore une fois. Cette nouvelle révolution trouve sa puissance dans la capacité d'effectuer des milliards de calculs par seconde. Cette capacité, dont l'importance est incomprise de la plupart des gens, est la clef du fonctionnement des services de transmission et de traitement de l'information, tels que les téléphones portables et les disques vidéo.

Chaque jour nous nous trouvons dans des situations qui demandent du calcul. On commence le matin, avec la planification de la journée. On continue au travail ou à l'école, en voyageant en voiture ou en train, et même parfois en calculant les calories de ses repas...

Le processus de calcul est souvent inconscient, par exemple, devant un produit qu'on voudrait acheter. Aujourd'hui le calcul direct et conscient, mental ou sur papier, joue un rôle très limité grâce à la présence des calculatrices, des ordinateurs, etc. Les moyens de calcul électronique ont remplacé les algorithmes de calcul manuel. Qui se rappelle comment extraire une racine carrée? Qui peut utiliser une règle à calcul? Qui peut encore résoudre la position d'un bateau en mesurant la hauteur d'un astre et en utilisant les tables trigonométriques? Toutes ces opérations sont faites avec des moyens électroniques. Parfois, les *digital assistants* présentent les problèmes sous une forme tellement différente de celle utilisée pour le calcul manuel qu'on ne s'aperçoit pas de la présence et de la complexité du calcul. Par exemple, il suffit d'appuyer sur un bouton pour retrouver sa propre position avec un récepteur de navigation par satellite *GPS*.

On pourrait dire que l'*homo sapiens* a été remplacé par l'*homo computans*. Aujourd'hui il n'est plus nécessaire de savoir beaucoup, mais il est nécessaire d'avoir

des moyens d'acquisition rapides d'information. Devant ces changements rapides et profonds causés par la diffusion de l'informatique et des moyens de calcul électronique dans la vie quotidienne, il est important de comprendre les origines historiques du calcul. Connaître l'évolution du calcul aide à le maîtriser et à comprendre les actions que, consciemment ou inconsciemment, nous faisons tout le temps.

Pour ces raisons, *Du zéro à l'ordinateur* est un livre que tout le monde devrait lire, pour apprécier la dimension historique d'une activité fondamentale et quotidienne. Il couvre, brièvement, cinq mille ans d'histoire, partant des notations et des méthodes de calcul de l'Egypte ancienne, pour en arriver aux puces électroniques.

Pour le lecteur commun, ce livre ouvre de nouveaux horizons sur l'informatique, et démontre comment les systèmes algorithmiques de calcul des anciens, comme la méthode d'Euclide du *plus grand commun diviseur*, forment la base des grands systèmes de calcul. Pour le scientifique, ce livre rappelle que les inventions récentes ne sont que l'évolution des techniques qui se sont développées à travers des siècles d'histoire dans différentes parties du monde.

Je souhaite aux lecteurs de retrouver dans ce livre le plaisir que j'ai eu en découvrant les racines et l'évolution du calcul.

Giovanni di Micheli
Stanford University

SOMMAIRE

AVANT-PROPOS

«Savoir comment un peuple compte,
c'est aussi savoir ce qu'il est.»
Charles Morazé

L'objet de cette histoire des chiffres, des machines à calculer et des ordinateurs est d'exposer la façon dont sont nées les idées qui ont conduit à la réalisation des ordinateurs et des microprocesseurs, dont nous faisons aujourd'hui un usage courant, et d'offrir au lecteur les clés qui lui permettront de mieux comprendre la nature et le fonctionnement de ces machines.

L'histoire des chiffres et des machines débute avec l'apparition des systèmes de numération.

Et comme le rappelle le titre du fameux livre de Samuel Noah Kramer, *L'histoire commence à Sumer*, c'est en effet dans cette région que furent retrouvés les plus anciens nombres écrits sur des tablettes d'argile, datant de 3000 ans avant Jésus-Christ. C'est là également que fut inventée l'écriture.

Depuis toujours, les hommes ont éprouvé le besoin de calculer afin de mesurer le temps, d'effectuer des comptes liés à l'agriculture ou au commerce, de confectionner certains objets ou de construire des maisons. Ils ont pour cela imaginé des systèmes de numération, puis des instruments de calcul (tels que les doigts) et divers instruments mécaniques, avant d'inventer des machines électroniques, les ordinateurs.

Rien ne permet d'affirmer qu'un point final ait été atteint dans cette évolution.

CHAPITRE 1

HISTOIRE DES CHIFFRES

«Dieu a créé les nombres entiers,
et l'homme a inventé le reste.»
Leopold Kronecker

Cette histoire du calcul débute par l'histoire des chiffres, en s'inspirant de la merveilleuse Histoire universelle des chiffres *de Georges Ifrah.*

Compter

On possède quelques informations quant aux premiers systèmes imaginés par l'homme pour compter les éléments d'un ensemble, tels que le nombre de bêtes constituant un troupeau.

L'un des premiers systèmes consistait à associer autant de nœuds sur une corde que de têtes de bétail. En associant un nœud à chacun des animaux passant devant soi lors de leur retour à l'étable, il était aisé de détecter une bête manquante. Mais la difficulté majeure résidait dans la confection même de la corde à nœuds, où des erreurs pouvaient survenir. Un autre système consistait à pratiquer des entailles sur un bâton ou un os, mais le problème demeurait là aussi de compter ces entailles. L'homme, comme l'animal supérieur, possède des capacités de perception limitées lorsqu'il s'agit d'estimer un nombre d'objets supérieur à 4 ou 5 éléments. Impossible pour lui, par exemple, d'évaluer le nombre exact de cailloux dans un tas. Il lui faut pour cela les compter.

La pratique des encoches sur les parois de cavernes à côté de dessins de divers animaux constitue la toute première comptabilité, consistant à enregistrer et à compter les animaux selon leur genre. L'entaille est réalisée partout avec à peu près la même méthode, et il est peu surprenant que ce système soit adopté tant en Europe qu'en Asie, en Afrique ou en Amérique. Partout ou presque, le «un» se symbolise par un trait vertical.

L'usage d'encoches pratiquées sur des os date de près de 20 000 ans. Dès cette époque, on observe une répartition par groupes de ces encoches lorsque leur nombre dépasse quelques dizaines. On voit donc déjà là une décomposition selon le principe des bases, lequel sera le fondement des systèmes de numération à venir.

Pour dénombrer des objets, le premier stade est pictural. On représente par exemple 4 dessins de vaches pour indiquer un troupeau composé de 4 vaches. Le stade suivant est symbolique: le même troupeau est alors représenté par un dessin d'une vache suivi de 4 traits. Ces quatre traits représentent le nombre 4, devenu alors une entité plus abstraite et pouvant qualifier tout ensemble d'objets [2].

Les premiers systèmes de numération

Impossible, en comptabilité, de tenir des livres de comptes de façon «orale». C'est pour cette raison que naquirent l'écriture et les chiffres, à Sumer, entre le Tigre et l'Euphrate [11, 12, 13]. Les premières tablettes d'argile qui y ont été découvertes sont des comptes agricoles, comportant à la fois des pictogrammes et des chiffres.

Les premiers systèmes de numération sont dits «additifs». Ils proposent des représentations de un à quatre comportant un à quatre signes, comme les chiffres romains I, II, III, IIII, avec un nouveau symbole V pour le cinq. On fait de même avec les chiffres suivants, VI, VII, VIII, VIIII et un nouveau symbole X pour le dix. Un grand nombre de civilisations ont adopté ce schéma dans leur système (Sumer, Egypte, Grèce, Mésopotamie, les Mayas, les Phéniciens) plus de 1000 ans avant notre ère. Une telle numérotation additive devient cependant vite compliquée avec de très grands nombres.

Le système de numération additif de Sumer

Ce système est le plus ancien et date d'environ 3200 avant J.-C. Il utilise des symboles différents pour les unités et les dizaines, et des symboles particuliers pour représenter 60, 600, 3600, 36 000 et 216 000 (fig. 1.1). C'est donc un système reposant sur la base 60 [1], et comportant deux formes, l'une dite archaïque, et l'autre plus récente dite cunéiforme. Pour former des nombres, on dessine autant de symboles que nécessaire (fig. 1.2).

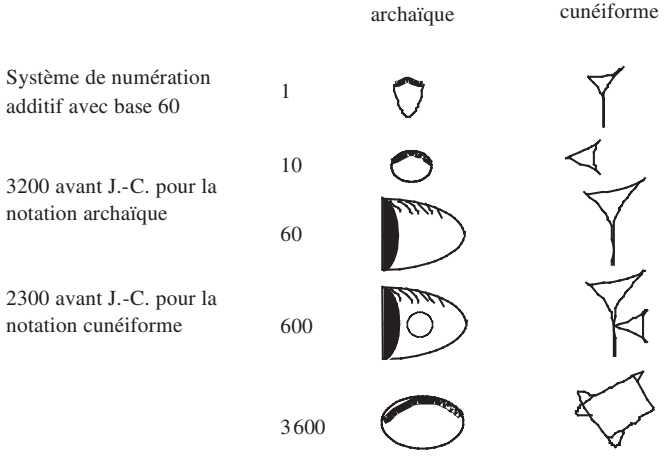


Fig. 1.1 Système de numération additif en base 60 de Sumer.

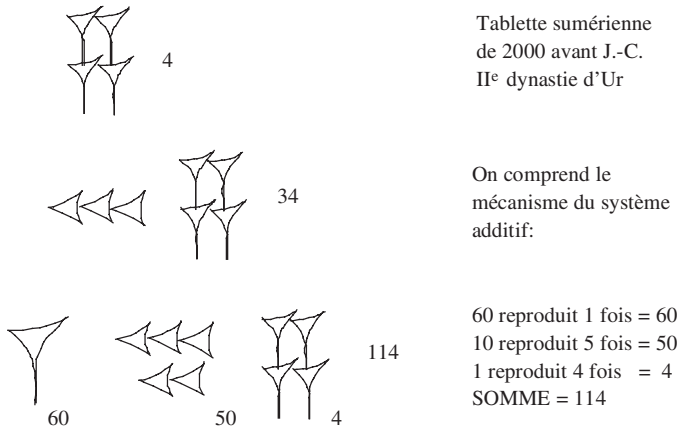


Fig. 1.2 Système additif de Sumer.

Les Sumériens sont les seuls dans l'histoire à avoir inventé un tel système de numération sexagésimal, mais les raisons du choix d'une base si élevée demeurent obscures. Le fait que l'année comporte près de 360 jours, ou que le nombre 60 ait été attribué au Grand Dieu Anu constituent les principales hypothèses.

On constate cependant que les symboles 10 et 600 (empruntés à la base 10, 600 représentant 60×10 et non 60×60) coexistent avec les symboles de la base 60 (1, 60, 3600, 216 000). La base 10 peut donc être considérée comme une base auxiliaire. On sait aussi qu'une base 12 coexistait à Sumer avec la base 60; cette dernière pourrait donc être la combinaison des bases 10 et 12.

Cette notation persista en Mésopotamie sous domination de Babylone, mais elle se transforma peu à peu en ajoutant des symboles correspondant à la base 10 (un symbole pour le 100, un autre pour le 1000), et mena à la coexistence de deux manières de représenter les nombres, c'est-à-dire en base 60 ou en base 10.

Le système de numération additif égyptien

L'écriture égyptienne, basée sur les hiéroglyphes, date de plus de 3000 ans avant Jésus-Christ. On la lit généralement de droite à gauche, le sens de lecture étant donné par l'orientation des oiseaux ou des têtes des personnages [11]. Cette écriture et le système de numération associé se sont développés indépendamment de ceux de Sumer, en adoptant notamment une base décimale pour le calcul.

Le système de numération additif égyptien [2] est apparu peu de temps après l'invention de l'écriture. Il a d'abord existé sous forme de hiéroglyphes, puis sous forme hiératique (sacrée), et enfin sous forme démotique (populaire).

☞ La figure 1.3 illustre ce système basé sur les hiéroglyphes. Le nombre 1 est un simple bâton (ou encoche). Le nombre 10 est représenté par un symbole pouvant être un seau ou un sac renversé contenant 10 objets. Le nombre 100 est une corde enroulée, probablement longue de 100 unités. Le nombre 1000 est une plante de lotus. Il existe encore des symboles différents pour représenter 10 000 (un doigt pointant le ciel), 100 000 (grenouille) et 1 000 000 (homme agenouillé). L'écriture d'un nombre est très régulière mais devient vite fastidieuse quand le nombre est grand. Celui-ci doit être lu de droite à gauche, bien que l'on trouve parfois quelques nombres écrits de gauche à droite.

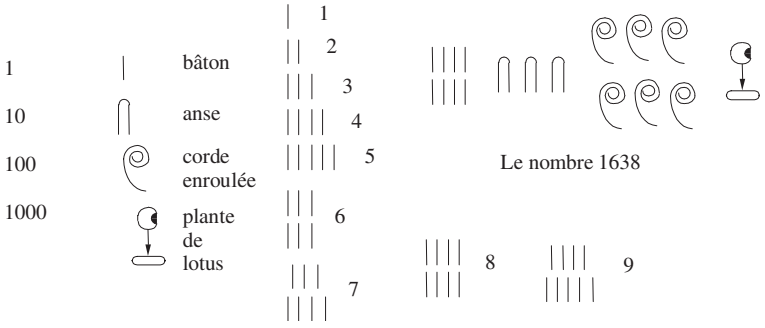


Fig. 1.3 Les nombres égyptiens en hiéroglyphes.

L'écriture hiéroglyphique était principalement utilisée pour les inscriptions sur des monuments.

Celle sur papyrus, dite hiératique, fut beaucoup plus utilisée, car d'un usage plus aisé. L'emploi de cette écriture cursive, dont les symboles sont très différents des hiéroglyphes, a subsisté fort longtemps pour la rédaction de textes religieux, usage dont elle tire son nom. Une autre écriture cursive, dite démotique, est quant à elle apparue vers le XII^{e} siècle avant Jésus-Christ, et employée principalement pour les usages courants.

☞ Dans ce système, l'addition s'effectue en regroupant tous les chiffres de même catégorie, puis en remplaçant dix signes identiques par un seul signe de la classe décimale supérieure. Pour la multiplication, on utilise une méthode de doublement successif. Prenons par exemple la multiplication de 12 par 12. Avec nos chiffres, on place le multiplicateur dans la deuxième colonne, la première colonne indiquant par quel nombre il est multiplié. On s'arrête dès que le doublement produit un résultat plus grand que le multiplicande, ici à 8 par 12 qui donne 96. Comme on veut multiplier par 12 et non pas par 8, on cherche les résultats partiels qui correspondent à la multiplication de 12 – 8 = 4. Or on sait que 4 fois 12 a déjà été calculé; il faut donc additionner ces deux résultats partiels pour obtenir le résultat final.

	1	12		
	2	24		
>	4	48	v	
	8	96	v	soit 96 + 48 = 144

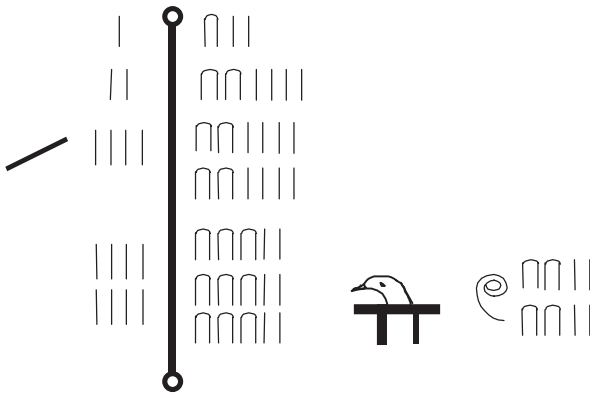


Fig. 1.4 Multiplication avec le système égyptien basé sur les hiéroglyphes.

La figure 1.4 représente la multiplication 12 par 12 avec les chiffres hiéroglyphiques égyptiens. Cet algorithme de multiplication par doublements successifs est, malgré son grand âge, le même que celui utilisé aujourd'hui encore dans les ordinateurs dépourvus de multiplicateur réalisé en matériel. Cet algorithme est connu sous le nom d'additions et décalages. Il consiste pour chaque pas de la multiplication à décaler à gauche le multiplicande et à l'additionner ou non selon le chiffre du multiplicateur. Un décalage à gauche du multiplicande revient à le doubler (fig. 1.15).

Le système de numération additif grec

Les Grecs utilisaient deux systèmes de notation de chiffres [2].

Le premier, très semblable au système égyptien, est le système attique, dont l'usage était en vigueur à Athènes. Il a pour particularité que, le «un» mis à part, ses chiffres sont les initiales des noms grecs correspondants. Le chiffre 5 est Pi, initiale de Pénté signifiant 5 en grec, le chiffre 10 est Delta, initiale de Déka signifiant 10. Un tel système est dit acrophonique.

Le deuxième système est celui des chiffres alphabétiques, illustré par la figure 1.5, qui utilise un très grand nombre de symboles, et permet par conséquent de coder un nombre avec peu de symboles. 96 est par exemple codé avec 15 symboles dans le système égyptien, alors que 2 symboles suffisent dans le système grec. Ce système présente l'avantage d'une économie de symboles, mais exige toutefois de mémoriser un grand nombre d'entre eux.

De tels systèmes ne sont plus plus en usage aujourd'hui, mais le principe additif demeure utilisé pour nommer les chiffres et les écrire avec nos lettres [1].

☞ Les dix premiers chiffres ont leurs propres noms, de un à neuf, puis dix, et l'on forme ensuite les suivants par addition, soit 1 + 10, un-dix, proche de onze (*undecim* en latin), puis 2 + 10, deux-dix, soit douze. A partir de dix-sept, la combinaison 7 + 10 est explicite, même si elle se dit à l'envers. Pour obtenir trente, quarante, etc., on procède par multiplication, soit 3 fois

10, 4 fois 10, etc. Seul le français parlé en France demeure particulier avec l'usage de «soixante-dix» ou «quatre-vingt-dix» au lieu de septante ou nonante. Reste le problème de vingt (on ne dit pas «duante»). Il faut remonter très loin, au sanskrit, langue indo-européenne, dans laquelle vingt se disait «vimsati» (soit dvi-sati, deux dizaines), proche du latin «viginti» et de «vingt» en français, pour en découvrir l'origine.

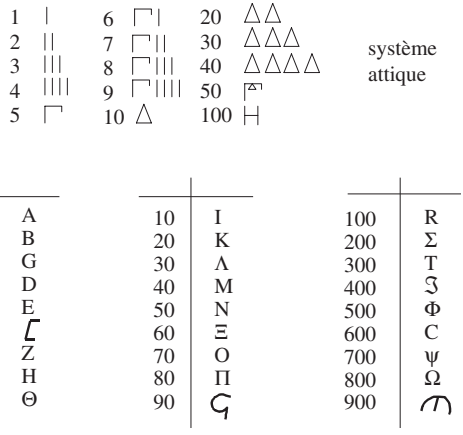


Fig. 1.5 Les chiffres attiques et alphabétiques des Grecs.

Le système de numération additif romain

Un autre exemple de système de numération additif est celui basé sur les chiffres romains. Comme ceux des Etrusques, ces chiffres puisent leur origine dans la pratique des entailles sur des bâtons.

Avec ce système, on écrit par exemple le nombre 77 généralement LXXVII, bien qu'il ne soit pas nécessaire de débuter par les symboles les plus grands [2]. Ecrire VIILXX ne serait pas ambigu, mais beaucoup plus difficile à lire.

La principale difficulté de ce système ne réside pas dans l'addition (fig. 1.6a), mais dans la multiplication, qui demande une décomposition préalable du multiplicateur. Ainsi, 13 se décompose en 10 + 2 + 1, ce qui revient à faire une addition avec 10 fois le nombre, 2 fois le nombre plus le nombre lui-même. On donne un autre exemple à la figure 1.6(b), avec la multiplication de 23 par 13.

Une autre règle veut qu'un chiffre placé à gauche d'un chiffre de valeur supérieure s'en retranche, comme IX pour le 9 ou XL pour 40. On vise ainsi à l'économie de symboles, bien que cela rende d'autant plus compliquée toute opération arithmétique. Cette notation existait déjà au II^{e} siècle après J.-C., mais son usage ne s'est répandu que vers 1600.

(a)
```
77   =  L    XX   V   II
+94  =  L    XXXX     IIII

171  =  LL   XXXXXX  V   IIIIII
```

On applique alors les règles IIIII = V, VV = X, XXXXX = L, LL = C, et l'on obtient:

```
171  =  LL   XXXXXX  VV  I
171  =  LL   XXXXXXX     I
171  =  LLL  XX         I
171  =  CL   XX         I
```

(b)
```
XX III              (×1)
XX III XX III       (×2)
CC XXX              (×10)
-----------------------------------------------
```

Total: CC XXX XX XX XX III III III, soit CC L XXXX V IIII, soit 299

Fig. 1.6 Addition (a) et multiplication (b) en chiffres romains.

Les bases

Si la base 10 s'est peu à peu imposée dans le monde entier, d'autres bases ont été longtemps utilisées, voire le sont toujours. C'est le cas, par exemple chez les peuples khmer et caraïbes, de la base 5 et dont l'origine est liée au cinq doigts de la main. Les Esquimaux, les Mayas et les Aztèques utilisaient la base 20 (comme les doigts des mains et des pieds additionnés) et son système associé, dit vigésimal. On trouve encore de nos jours des traces de ce système dans le «quatre-vingt» du français (seuls les Vaudois disent huitante), ou le «trois-vingts» et «six-vingts» utilisés autrefois pour 60 et 120.

Les langues danoise et basque nomment aussi certains nombres de cette manière.

De nos jours, on utilise encore la base 60 pour la mesure du temps (une heure comporte 60 minutes de 60 secondes) et celle des angles (un cercle est bizarrement partagé en 360°). Swatch a récemment lancé le nouveau standard Internet Time (@300 ou @456), dans lequel le jour est divisé en 1000 «beats», à l'instar du calendrier républicain qui, durant les 13 ans suivant la Révolution française où il fut en usage, introduisit des jours de 10 heures de 100 minutes.

Numération de position

Pour remédier aux problèmes de calcul posés par la numération additive, on imagina une autre numération, dite «de position», où la localisation d'un chiffre dans un nombre lui confère une valeur particulière. C'est ainsi que les unités, les dizaines ou les centaines furent inventées, sur le principe d'une base dix. Ainsi, 248 est composé de 2 centaines, 4 dizaines et 8 unités.

Ce système a été inventé trois fois, d'abord par les Babyloniens au 2^{e} millénaire avant Jésus-Christ, puis par les Chinois un peu avant le début de notre ère et enfin par

les Mayas [1]. Mais c'est à Mari, en Mésopotamie, que furent retrouvées les traces les plus anciennes de cette numération, qui datent du 18e siècle avant Jésus-Christ.

Le système positionnel de Mari

Ce système fut découvert en 1933, lorsque des fouilles sur le site mirent à jour près de 20 000 tablettes de comptes, utilisées comme listes de besoins pour la vie quotidienne. Comme le montre la figure 1.7, ce système utilisait le symbole du clou pour représenter à la fois les unités (soit de 1 à 9 clous pour les chiffres 1 à 9) et les centaines, et un autre symbole pour représenter les dizaines. Selon sa position, le clou prenait donc la valeur d'unité ou de centaine, caractéristique d'un système positionnel (l'usage d'un symbole différent pour les dizaines lui interdisant toutefois cette appelation *stricto sensu*).

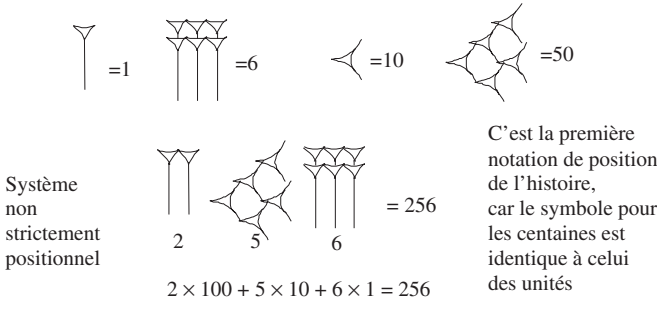


Fig. 1.7 Système non strictement positionnel de Mari (1800 avant J.-C.).

On pense que ce système est apparu de manière progressive. Si l'on se reporte au système antérieur non positionnel de Babylone (fig. 1.1 et 1.2), on constate que le petit clou représentait les unités et le grand clou les soixantaines. Il semble que, petit à petit, la valeur de la différence de taille s'est amenuisée au profit de la position du symbole, avant de disparaître complètement. Il est en effet probable que la similitude entre les symboles représentatifs des unités et des centaines utilisés dans le système de numération additif soit à l'origine de l'apparition du système de numération positionnel.

Le système positionnel de Babylone

Les Babyloniens utilisaient un système de numération positionnel à base 60 (fig. 1.8). Les symboles pour les dizaines et les unités étaient compris dans le même ordre, la base 10 constituant une base auxiliaire (il serait fastidieux de dessiner 41 ou 56 petits clous!). On utilisait le même ensemble de symboles pour représenter les unités (de 1 à 59), les soixantaines (de 1×60 à 59×60), et ainsi de suite.

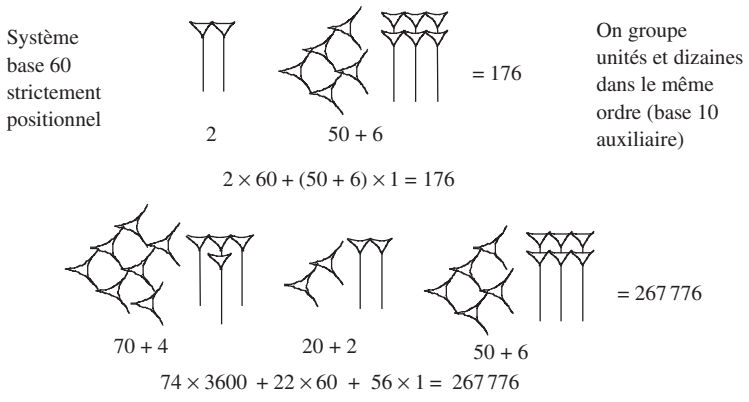


Fig. 1.8 Système positionnel base 60 de Babylone.

Une des difficultés de ce système résidait dans l'écriture de certains nombres, notamment ceux comportant un zéro en position intermédiaire (exemple: 206)

Les savants babyloniens ne connaissaient pas le zéro, ce qui constituait une difficulté majeure pour calculer [1]. En base 60, la soixantaine était représentée par un clou, identique au clou représentant l'unité. Ainsi, pour représenter 60, il aurait fallu faire suivre ce clou de quelque chose d'analogue au zéro pour comprendre que l'on parlait de soixantaines et non d'unités. Mais un clou isolé pouvait signifier 1, 60 ou 3600! On connaît un certain nombre d'exemples de nombres représentés sur une tablette qui sont ambigus (fig. 1.9).

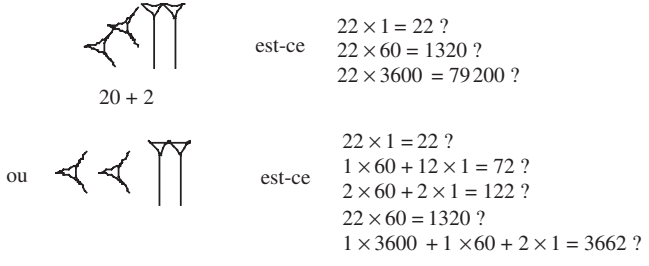


Fig. 1.9 Exemple de difficultés d'interprétation du système positionnel de Babylone.

Dès le IV^{e} siècle avant Jésus-Christ, on utilise un espace ou un signe de séparation pour signifier l'absence d'un chiffre d'un rang donné: c'est le plus vieux zéro de l'histoire. Mais insérer un espace peut être source d'erreur, car un scribe peu au fait de ces difficultés ne le retranscrira pas, et changera sans s'en apercevoir la valeur du nombre.

Un signe particulier est introduit vers 300 avant J.-C. pour représenter le zéro (fig. 1.10) mais il n'est utilisé qu'en position médiane et non terminale, et ne résout donc pas les problèmes posés par le système. Ce zéro n'est pas utilisé en tant que nombre zéro mais comme symbole représentant un vide.

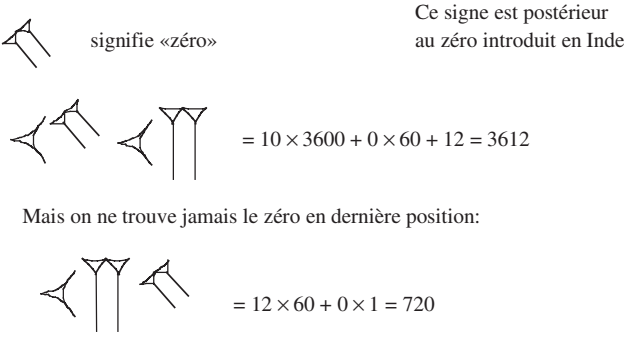


Fig. 1.10 Un signe introduit comme «zéro».

Le système positionnel chinois

La notation positionnelle chinoise date de 540 avant J.-C., certaines données remontant même à 1300 avant J.-C. [2]. Les chiffres 1 à 5 sont symbolisés par 1 à 5 bâtons, les chiffres 6 à 9 par 1 à 4 bâtons surlignés. Les Chinois inventent un système ingénieux pour fixer le rang du chiffre. Les bâtons sont verticaux pour représenter les unités, les centaines, les dizaines de milliers, etc, et horizontaux pour représenter les dizaines, les milliers et les centaines de milliers (fig. 1.11). Ainsi, si deux groupes de symboles sont verticaux, c'est qu'il y a un espace symbolisant un «zéro» entre les deux. Mais lorsque l'on doit, par exemple, représenter 4000, l'ambiguïté subsiste.

Des damiers fixant les positions des chiffres permirent de laisser un espace vide pour symboliser le «zéro», là encore non opératoire. Celui-ci fut remplacé par un petit rond vers 800 après J.-C., mais on sait que ce «zéro opératoire» fut probablement introduit après divers contacts antérieurs avec l'Inde.

Le système positionnel maya

Les Mayas, au 1er millénaire de notre ère, utilisent une numération de position en base 20. Ils inventent le zéro entre le IVe et le IXe siècle après Jésus-Christ en le symbolisant par un petit coquillage. Mais par une curieuse irrégularité dans leur base 20 (au lieu de 1, 20, 400 et 8000, ils utilisent 1, 20, 360 et 7200), le zéro n'est pas opératoire et ne permet pas de calculer.

Ce particularisme peut s'expliquer par les 360 jours que comportait l'année maya. Comme pour les Babyloniens, le zéro représentait ici un espace vide et non un signe opératoire.

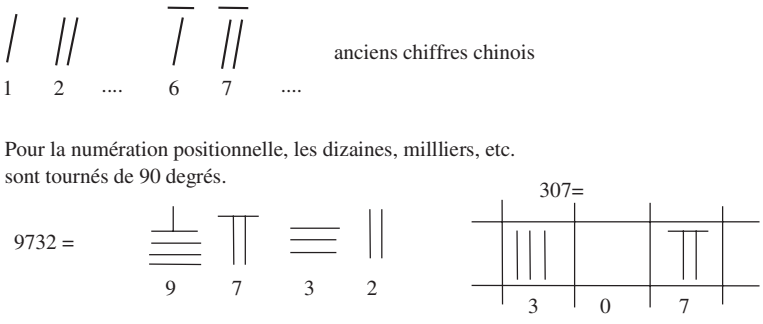


Fig. 1.11 Numération positionnelle chinoise.

Un tel système fut utilisé pour le calendrier dit «long compte» des Mayas classiques [3], dont les mythes rapportent l'apparition vers 3113 avant Jésus-Christ, et dont l'usage ne devint courant que lors des premiers siècles de notre ère.

☞ La figure 1.12 illustre une date de ce calendrier. Les différents dessins sont appelés des glyphes. Le premier glyphe est introducteur; il indique que l'inscription qui suit est une date «long compte». La numérotation de position est ensuite utilisée:

- premier glyphe «baktun» (en-dessous à gauche), le grand cycle de 144 000 jours, soit 20 fois 7200 jours. On y voit une barre et quatre points, soit 5 + 4 = 9 baktun, soit 9 × 144 000 jours
- deuxième glyphe «katun» (à la droite du premier), un cycle de 7200 jours. On y voit deux barres et quatre points, soit (2 × 5) + 4 = 14 katun, soit 14 × 7200 jours
- troisième glyphe «tun» (en dessous à gauche), un cycle de 360 jours. Il y a 3 barres et à nouveau 4 points, soit (3 × 5) + 4 = 19 tun, soit 19 × 360 jours
- quatrième glyphe «uinal» (à la droite du premier), un cycle de 20 jours. Il y a une seule barre et trois points, soit 5 + 3 = 8 uinal, soit 8 × 20 jours
- enfin, en dessous à gauche, le dernier glyphe «kin» (un jour) de cette date comporte un signe en forme de papillon qui signifie «zéro».

On peut donc lire cette date comme suit:

9 × 144 000 jours + 14 × 7200 jours + 19 × 360 jours + 8 × 20 jours + 0 jours = 1 403 800 jours (environ 3846 ans) à compter depuis 3113 avant notre ère, ce qui donne environ 730 après Jésus-Christ.

Le système positionnel aztèque

Le système positionnel aztèque est cohérent par rapport à celui des Mayas, et utilise le 1, 20, 400 et 8000 [4]. La position 20 est un compte (*cem-poalli*), la position 400 est une chevelure (*cen-tzontli*) et la position 8000 est un sac (*ce-xiquipilli*). Les différentes positions sont représentées par des idéogrammes, un rond pour 1, un drapeau pour 20, une plume ou un fagot pour 400 et un sac pour 8000 (fig. 1.13).

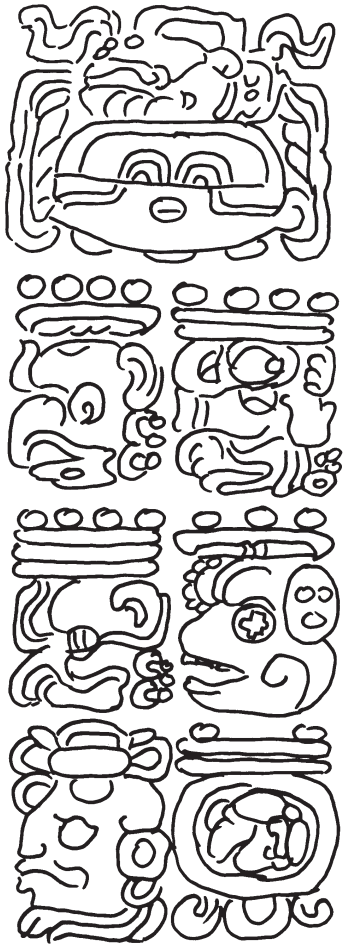


Fig. 1.12 Date dans le calendrier maya.

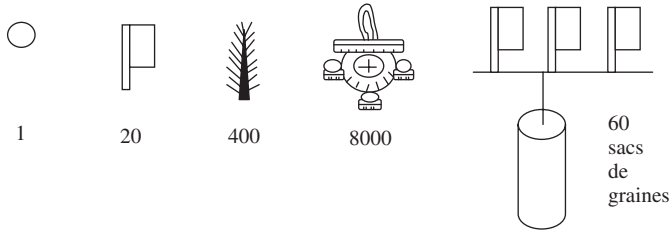


Fig. 1.13 Les chiffres aztèques.

Systèmes de numération de position avec zéro opératoire

Le système de numération indien

C'est à l'Inde que nous devons notre système de numération moderne. On le qualifie à tort de numération «arabe», car les Arabes n'ont joué qu'un rôle d'intermédiaires entre l'Inde et l'Occident.

Le plus ancien document faisant état de calculs avec le zéro date de 458 de notre ère: il s'agit d'un traité de cosmologie, le *Lokavibhâga*, soit «Les parties de l'Univers». Ce document prouve qu'un système de numération de position avec zéro opératoire est en usage en Inde dès le IV^{e} siècle, au beau milieu de l'époque classique indienne de l'empire des Gupta (240-535), renommée pour sa très haute expression culturelle et artistique.

La figure 1.14 représente des chiffres indiens 0 à 9 et montre une similitude avec nos chiffres arabes (au moins pour 1, 2 et 3). On pense que la graphie de ces chiffres a évolué à partir de l'ancienne notation consistant en un trait pour le un, deux traits pour le deux, etc.

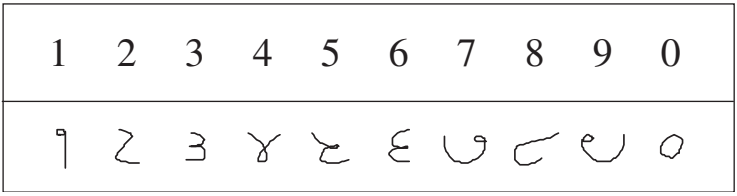


Fig. 1.14 Ecriture des chiffres indiens dans la région de Bombay.

Un des avantages majeurs de ce système est qu'il permet d'exprimer de très grands nombres, en ajoutant à droite autant de zéros que désiré. En Inde, un nom avait ainsi été donné à chaque ordre, jusqu'au 18^{e} (1 suivi de 18 zéros). Les Romains n'avaient pas la possibilité d'exprimer des nombres au-delà de 100 000 et il fallut attendre l'an 1270 pour que le français introduise le mot «million» à la suite des 3 autres ordres (mille, dix-mille, cent-mille). Au milieu du XVII^{e} siècle apparaissent les termes de billion, trillion, quadrillion, etc, signifiant respectivement 10^{9}, 10^{12}, 10^{15}, jusqu'au nonillion égal à 10^{30}. L'anglais a conservé le billion pour 10^{9}, alors que le français parle de milliard et que l'actuel billion est un million de milliards, soit 10^{12}, le trillion un million de billions, soit 10^{18}. Le quadrillion est quant à lui absent du *Petit Larousse*. Notons que le plus grand nombre possédant une signification physique est 10^{42}, soit le rapport entre les diamètres de l'Univers et d'un atome, égal au nombre de protons et de neutrons dans l'Univers.

Le système de numération arabe

Nombreux sont ceux qui croient que les chiffres ainsi que le zéro sont dus aux Arabes. L'introduction dans le monde arabe de la numération de position indienne n'a toutefois eu lieu qu'entre 750 et 800, au début de l'âge d'or de la science arabe, comme en témoigne la visite de savants indiens en terre d'Islam en 776. Un autre système basé sur les lettres arabes a toutefois été utilisé simultanément. Quelques années plus tard, le grand savant arabe Al-Khuwarizmi (783-850) écrit un livre intitulé Al Jabr. *Ce titre, ainsi que le nom de ce savant, sont respectivement à l'origine des mots «algèbre» et «algorithme».*

L'introduction en Europe de la numération indienne prit quelques siècles. L'instigateur en fut Léonard de Pise (1175-1250), connu sous le nom de Fibonacci. Il est l'auteur du *Livre de l'Abaque*, dont le succès fut très limité – vraisemblablement à cause des 459 pages fastidieuses à copier qu'il comporte – dans lequel il décrit les chiffres arabes. En 1299, la Ville de Florence interdit l'usage des chiffres arabes, parce l'on pouvait facilement transformer un 0 en un 9 ou un 6.

Une première introduction, sans le zéro, eut lieu au X^{e} siècle par le moine Gerbert d'Aurillac, qui deviendra pape en l'an mille. On n'utilisait alors les chiffres arabes que pour le calcul sur abaques, l'usage des chiffres romains restant très majoritairement prédominant.

La numération arabe, cette fois accompagnée du zéro, réapparut au XII^{e} siècle, dans la foulée des croisades. Elle se heurta une fois encore à une vive hostilité de la part des détenteurs du savoir du calcul sur abaques, ainsi que des membres de l'Eglise, qui qualifiaient cette invention «arabe» de maléfique. Son introduction fut si lente qu'en 1575, Montaigne, fort de sa célèbre devise «Que sais-je?» écrit qu'il ne sait pas calculer (chap. 2).

L'abaque à calcul, appelée aussi échiquier, a ainsi subsisté longtemps encore. Le ministre des finances anglais porte aujourd'hui encore le titre de «Chancelier de l'Echiquier».

Le système de numération de position avec le zéro est un système parfait et universel, contrairement à la tour de Babel de nos multiples langues. Il a permis le développement des mathématiques ainsi que celui des machines à calculer et des ordinateurs, bien que ceux-ci travaillent en base 2 et non 10. Ce fut George Boole (1815-1864) qui développa la logique symbolique avec les opérateurs ET, OU et NON, basée sur deux symboles «vrai» et «faux», correspondant aux deux chiffres «1» et «0» de la numération en base 2.

Système de numération en base 2

Dès 1703, Leibnitz montre la très grande simplicité du système binaire qui n'emploie que deux symboles «0» et «1». Les opérations arithmétiques sont en effet extrêmement simples en base 2, et par conséquent idéales pour leur automatisation sur ordinateurs. Les dispositifs dont sont formés les ordinateurs sont des interrupteurs

(appelés transistors) qui présentent deux états, «ouvert» ou «fermé», symbolisés par le «0» et par le «1». La base 2 permet donc de réaliser des unités de calcul en base 2 à la fois simples, robustes et peu coûteuses.

☞ La figure 1.15 illustre une manière de réaliser une multiplication en base 2. On utilise l'algorithme d'additions et décalages en vigueur en Egypte vers 1700 avant Jésus-Christ, en multipliant le multiplicande 5 (0101 en binaire) par le multiplicateur 13 (1101 en binaire).

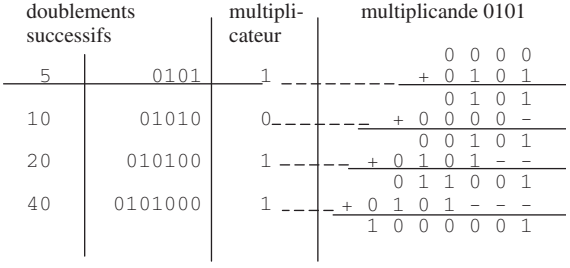


Fig. 1.15 Multiplication en binaire.

Les deux premières colonnes représentent le doublement successif du multiplicande. Ces quantités doivent être ajoutées ou non selon que le bit du multiplicateur (3e colonne, le bit de poids faible en haut) est «1» ou «0». Ainsi, le premier bit de poids faible du multiplicateur étant «1», on ajoute (4e colonne) le multiplicande. Le 2e bit étant «0», on n'ajoute rien. Le 3e bit étant à nouveau «1», on ajoute la quantité «0101--», soit 010100 ou 20 en décimal, soit le multiplicande décalé de 2 positions, correspondant à deux doublements successifs. On procède de même pour le 4e bit du multiplicateur qui est aussi «1», et l'on ajoute la quantité «0101000» décalée d'une position qui est 40 en décimal, le double de 20. On obtient bien le résultat en binaire qui est 5 + 20 + 40 = 65 en décimal.

Système de numération redondant

Inventé il y a quelques dizaines d'années [7], ce système n'est employé qu'en informatique. Il repose sur le principe selon lequel un même nombre peut être représenté de plusieurs manières. Un exemple illustrant deux manières de représenter 1999 est donné à la figure 1.16.

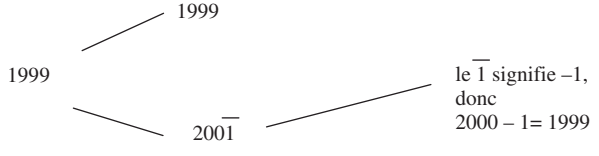


Fig. 1.16 Exemple de numération redondante.

D'après cet exemple ci-dessus, $200\,\overline{1}$ comporte davantage de zéros que 1999. Multiplier par $200\,\overline{1}$ devient alors plus simple que de multiplier par 1999. L'addition se simplifie elle aussi. Pour les ordinateurs, la séquence d'opérations s'effectuant depuis les unités vers les dizaines est une opération lente, compliquée par les reports. L'utilisation d'un système de numération redondant permet d'accélérer considérablement l'opération. L'exemple suivant montre toutefois que le système présente quelques subtilités (fig. 1.17).

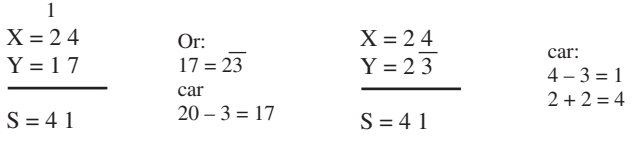


Fig. 1.17 Addition sans report.

La figure 1.17 illustre une addition normale (24 + 17), nécessitant un report des unités sur les dizaines. Afin d'éviter le report, on peut représenter le nombre 17 en utilisant une numération redondante, soit $2\,\overline{3}$. Ainsi, au lieu d'ajouter 17 à 24, on ajoute $2\,\overline{3}$ à 24: ni la soustraction des unités 4 – 3 ni l'addition des dizaines 2 + 2 n'occasione alors de report. L'usage d'une représentation redondante est particulièrement utile lorsque l'opération comporte des nombres aux chiffres élevés, puisqu'ils produisent inévitablement des reports.

La symbolique des nombres

De tout temps, les hommes ont éprouvé le besoin d'attribuer aux chiffres des propriétés magiques ou des significations ésotériques. On retrouve une multitude de significations dans des manuscrits anciens [5].

La matière, par exemple, est régie par le nombre 4, associé aux quatre éléments air, feu, terre et eau. Le centre du monde est la croix, avec ses quatre branches aux quatre coins cardinaux. L'urbanisme du Moyen Age adopte ce modèle, avec un centre-ville où se situe l'Eglise ou la Cathédrale, et deux rues en croix vers les quatre points cardinaux. Nous trouvons aussi l'Unité, symbolisant Dieu, le binaire, qui représente l'Homme d'un côté et l'Univers de l'autre, le ternaire, qui unit les trois premiers (1 + 2 = 3), et le dénaire, qui représente la perfection (1 + 2 + 3 + 4 = 10). Le roman a repris ce type d'interprétations [10]. Le nombre «onze» transgresse «dix», celui des Commandements, et symbolise le péché [15].

Les principaux concepts

On peut résumer les concepts de manière relativement simple:

- Les premiers systèmes de numération sont additifs, leur principe consiste à représenter autant de symboles qu'il y a d'unités ou de dizaines à dénombrer.
- les systèmes de numération de position, mais sans zéro opératoire, comportent en général des ambiguïtés, en particulier si le ou les zéros doivent être placés à la fin du nombre.
- les systèmes de numération de position avec zéro opératoire, découverts en Inde il y a plus de 15 siècles, constituent le système parfait, et bien sûr celui en usage aujourd'hui en base 10 (et en base 2 pour les ordinateurs). Si 4000 langues existent encore aujourd'hui sur Terre, un seul et unique système de numération s'est imposé.
- un système de numération différent a été inventé il y a quelques dizaines d'années, le système redondant, permettant aux ordinateurs d'effectuer des additions sans report.
- La croyance selon laquelle les Arabes auraient inventé la numérotation de position et le zéro reste vivace [6]. Certains l'attribuent même aux Mayas, «ce peuple merveilleux qui inventa le zéro et l'infini» (relevé lors d'une exposition à Venise en septembre 1998).

Références

[1] Georges IFRAH, *Histoire universelle des chiffres*, Bouquins, Robert Laffont, 1994.

[2] M. R. WILLIAMS, *History of Computing Technology*, IEEE Computer Society Press, Los Alamitos, CA, second edition, 1997.

[3] *La Recherche* n° 82, octobre 1977, p. 870.

[4] H. HARVEY, B. WILLIAMS, «L'arithmétique aztèque», *La Recherche* n° 126, Octobre 1981, pp. 1068-1081.

[5] Jean-Pierre BRACH, *La symbolique des nombres*, Collection Que sais-je? PUF n° 2898, 1994.

[6] Mario de BLASI, *Computer Architecture*, Addison-Wesley, 1990, p. 26.

[7] M. J. IRWIN, R. M. OWENS, «Digit-Pipelined Arithmetic as Illustrated by the Paste-Up System: A Tutorial», *IEEE Computer*, April 1987, pp. 61-67.

[8] C. HOUZEL, «L'invention du zéro», *Pour la Science*, n° 228, oct. 1996, pp.12-16.

[9] M. ASCHER, *Mathématiques venues d'ailleurs*, Seuil/sciences ouvertes, 1998.

[10] B. WERBER, *Le père de nos pères*, Albin Michel, Paris, 1998, pp. 64-66.

[11] Georges JEAN, *L'écriture, mémoire des hommes*, 24/Découvertes Gallimard/archéologie, 1987.

[12] «L'univers des nombres», *La Recherche* n° 322, hors série, 2 août 1999.

[13] Samuel Noah KRAMER, *L'histoire commence à Sumer*, Champs, Flammarion, 1994.

[14] Alain SCHÄRLIG, *Compter avec des cailloux*, Presses polytechniques et universitaires romandes, 2001.

[15] B. MAITTE, «La Lumière», Collection Points Sciences S28, 1981, p. 27.

[16] A. WARUSFEL, «Les nombres et leurs mystères» Collection Points Sciences S21, Ed. Du Seuil, 1961.

[17] E. GARRISON, *A History of Engineering and Technology*, CRC Press, Second Edition, 1998, Chapter 2 «Early Empires and the conquest of materials», pp. 15-43.

CHAPITRE 2

HISTOIRE DES MACHINES À CALCULER ET DES AUTOMATES

«Aucune machine ne remplacera
un employé fiable et honnête.»
Le Président de Remington Company, 1897

Les instruments de calcul

*Littéralement, «calculer» signifie grouper des cailloux (*calculi *en latin) par dizaines ou par centaines afin de réaliser des opérations arithmétiques. Mais le premier instrument de calcul en vigueur fut sans doute la main, avant même que n'apparaissent l'os à encoches ou la corde à nœuds chez les Incas [14].*

Il existe en effet de très nombreux systèmes de comptage basés sur les doigts et utilisant une gestuelle permettant de compter jusqu'à plusieurs milliers. Les mots latins *digiti*, (doigts), et *articuli* (articulations), désignaient au Moyen Age les unités et les dizaines. Le mot *digit* signifie aujourd'hui chiffre en anglais, et à l'ère des ordinateurs qui calculent en base 2, on désigne les symboles «0» et «1» par «binary digit», ou «bit».

D'après Hérodote, une gestuelle utilisant les doigts était en usage en Grèce depuis le V^{e} siècle avant notre ère [4]. D'Aristophane à Plutarque, de nombreux auteurs grecs témoignent de cette pratique, tout comme Cicéron qui l'observe à Rome. La main est donc sans doute le plus ancien instrument de calcul jamais utilisé, et cette pratique trouve son origine vraisemblablement en Egypte [1].

L'objectif premier de cette gestuelle était de signifier un nombre à son interlocuteur, comme l'illustre la figure 2.1, issue du *Summa de arithmetica* rédigé à Venise en 1494 par Luca Pacioli. La main gauche permettait d'aller jusqu'à 90, tandis que la main droite allait de 100 à 9000. Dès le haut Moyen Age, la représentation des nombres par les mains permet aussi de mémoriser les retenues lors des opérations effectuées mentalement ou par l'intermédiaire d'abaques [16].

En 1202, Léonard de Pise témoigne de cet usage dans son ouvrage *Liber abaci*, où il propose de «garder en main» les reports dans les multiplications réalisées à l'aide de tables à poussière, sur lesquelles on peut aisément écrire et effacer des chiffres. Luca Pacioli propose lui aussi cet usage dans *Summa de arithmetica*, dont est extraite la gravure sur bois représentée à la figure 2.1

Mais si les doigts servaient avant tout à mémoriser les nombres et à les communiquer, le premier véritable instrument de calcul fut le caillou. Ce système basé sur une association entre forme et valeur permettait de réaliser des opérations arithmétiques de manière fort similaire à celui employant la notation archaïque de Sumer, apparue 3000 ans avant J.-C. (fig. 2.2).

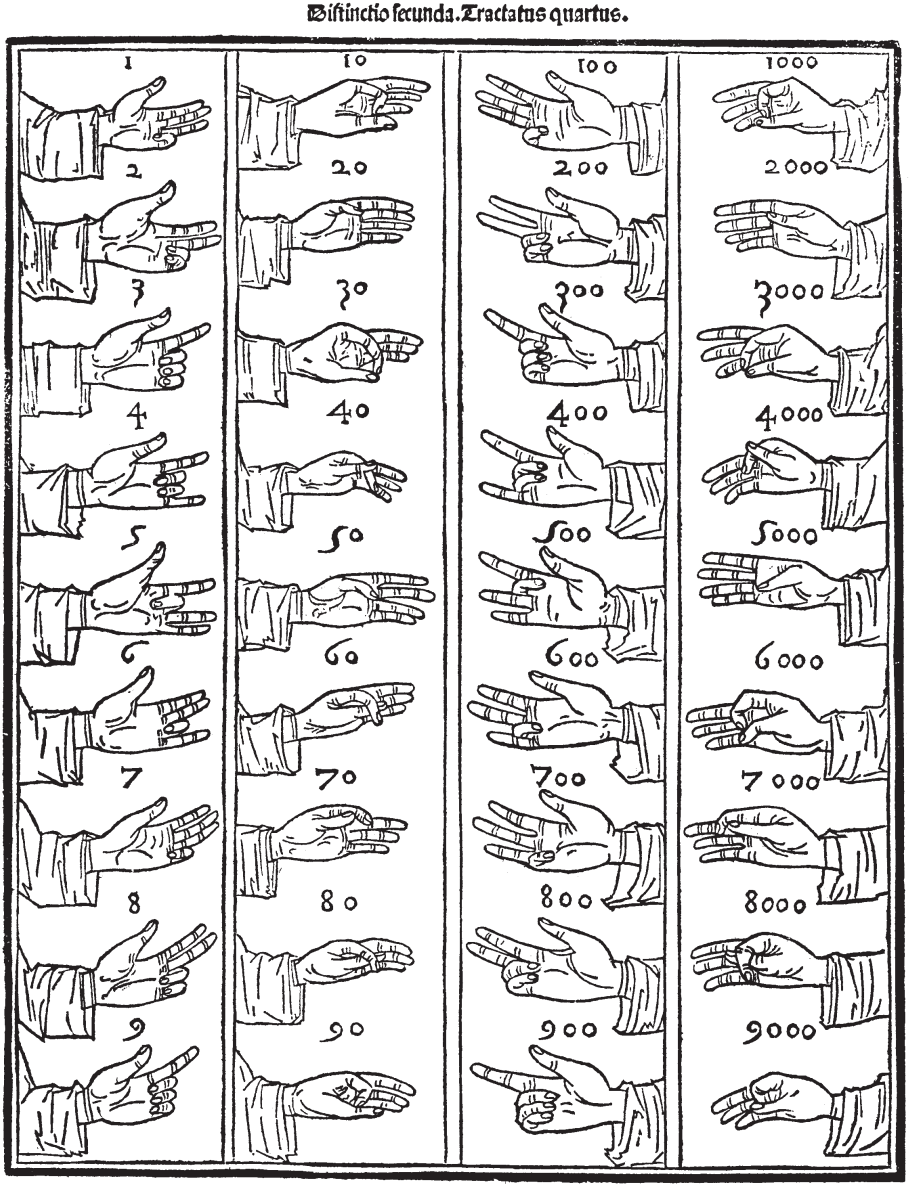


Fig. 2.1 Représentation de nombres avec les mains.
(Illustration issue du *Summa de arithmetica* de Luca Pacioli, 1494.)

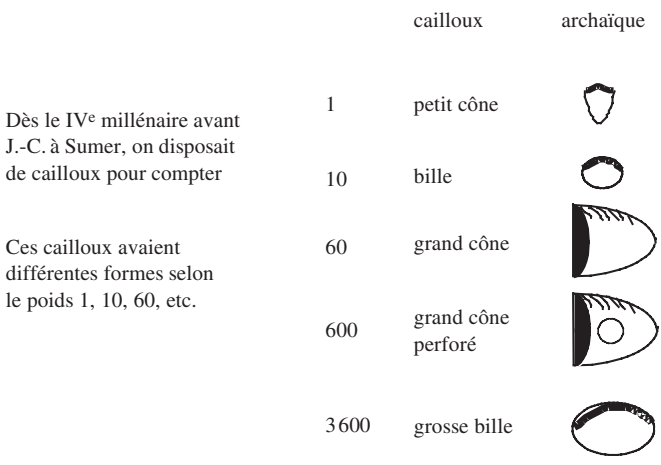


Fig. 2.2 Quelques cailloux utilisés à Sumer pour le calcul.

Les abaques

Lorsque les algorithmes de calcul apparaissent, leur diffusion se trouve limitée par le coût élevé du papyrus et l'absence de support bon marché tel que le papier [4]. C'est pourquoi d'autres instruments de calcul vont apparaître, moins primitifs que les simples cailloux, tels les abaques qui vont permettre d'appliquer certains algorithmes de calcul.

L'origine du mot abaque est sémite (*abaq*) et signifie poussière, à l'image de celle qui recouvrait les première surfaces que l'on utilisait pour calculer et que l'on marquait avec le doigt.

Les abaques apparaissent vraissemblablement pour la première fois en Grèce, avant d'être couramment utilisés en Chine vers 500 avant J.-C. [10]. Au 1er siècle de notre ère, l'abaque romain consiste en une plaquette de métal comportant des rainures dans lesquelles on place des jetons, chacun d'eux étant associé à un ordre d'unité. L'abaque comprend neuf rainures parallèles, plus une ou deux pour les fractions (fig. 2.3) et fonctionne sur un principe identique à celui des bouliers.

Chaque rainure est divisée en deux parties: une partie inférieure où les jetons représentent une unité, et la partie supérieure où ils valent 5 unités. L'ensemble des rainures est ordonné de droite à gauche en débutant par les unités symbolisées par (I), les dizaines (X), les centaines (C), et ainsi de suite jusqu'au [X], symbolisant un million. Le chiffre indiqué par l'abaque de la figure 2.3 est ainsi 1735. La rainure la plus à droite (e) permet de faire figurer des fractions, comme 1/2, 1/3 ou 1/4, valeurs définies selon la position des jetons.

☞ Pour réaliser une addition, telle que celle consistant à ajouter 18 à 1735, on ajoute un jeton valant 5 et 3 jetons valant 1 dans la rainure des unités. Cette somme étant alors égale à 13, on supprime les deux jetons valant 5 pour les remplacer par un jeton valant 1 dans la rainure

des dizaines. On trouve alors dans cette rainure, outre les trois jetons d'origine, ceux associés respectivement aux dizaines de 13 et de 18. Ces 5 jetons peuvent être remplacés par un jeton de valeur 5, donnant un résultat final égal à 1753.

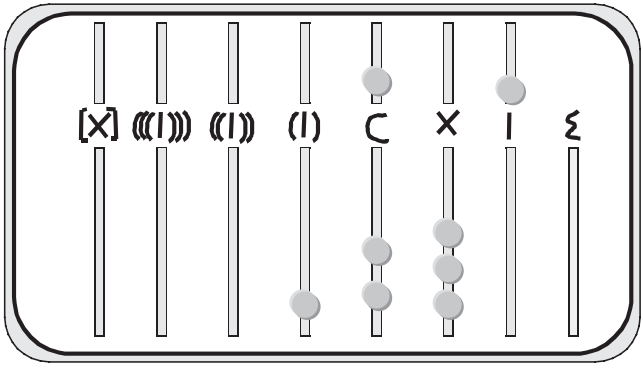


Fig. 2.3 Abaque romain du début de l'ère chrétienne.

L'autre exemple donné figure 2.4 est basé sur l'emploi d'un abaque différent du précédent. Les jetons sont dans ce cas des boutons mobiles que l'on fait glisser dans la rainure pour indiquer le nombre voulu. Pour qu'un jeton soit considéré dans le calcul, il faut qu'il soit placé près du centre de la rainure.

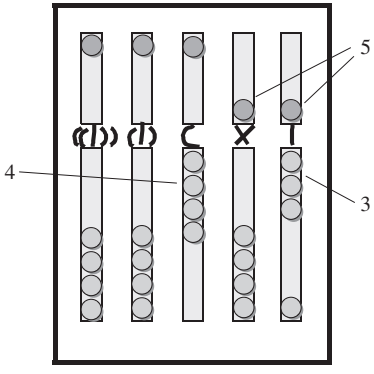


Fig. 2.4 Abaque romain. Le nombre illustré dans cette figure est égal à 458.

Il faut remonter au 1er siècle avant notre ère pour qu'Horace signale l'existence chez les Romains d'un autre instrument, l'abaque à cire. C'est une planchette de bois recouverte de cire, que l'on marque d'un stylet et qui est utilisée en guise de calculette portative.

Les abacistes et les algoristes

Une longue lutte débuta au XII^e siècle entre les abacistes, tenants du calcul avec les abaques, et les algoristes, partisans du calcul aux méthodes algorithmiques. Ce conflit se perpétua jusqu'à la Renaissance [16].

La discorde apparut avec la traduction en latin de traités d'arithmétique arabes présentant des algorithmes de calcul et d'algèbre originaires d'Inde et du monde arabe. Ces ouvrages utilisant les chiffres arabes proposaient des alternatives dans la manière d'écrire les chiffres 0 à 9, et montraient comment multiplier ou diviser en utilisant ces chiffres. L'enseignement de l'algorisme commença alors à être dispensé dans les premières universités, mais le calcul sur abaque résista encore fort longtemps. Considéré comme une arithmétique pratique, notamment pour les opérations d'usage quotidien, son usage fut popularisé par la publication d'ouvrages tels que *Arithmétique par les gects* (1558) et *Arithmétique avec l'art de calculer aux getons*, qui rencontrèrent un grand succès. Lorsque Montaigne décrit dans ses *Essais* la reprise du domaine familial à la mort de son père en 1568, il note: «Or je ne sais compter ni à jet ni à plume» [17], indiquant ainsi qu'il ne maîtrise ni le calcul sur abaque, ni le calcul à chiffre. Et bien que Léonard de Pise ait, dès 1202, recommandé l'usage des algorithmes de calcul pour la multiplication, le calcul sur abaque demeura usité jusqu'au XVII^e siècle. Leibnitz (1646-1716) l'utilisera encore notamment pour la réalisation de longs calculs [16].

Les bouliers

Le boulier ressemble beaucoup à l'abaque, mais son apparition est très récente.

Le boulier chinois, toujours en usage de nos jours, apparaît au XIV^e siècle de notre ère. Il comporte 7 boules par tige, dont 5 boules pour les unités et 2 boules de poids 5. On peut donc représenter un nombre de valeur maximale égale à 15 par tige pour des résultats intermédiaires (fig. 2.5).

Pour indiquer par exemple une valeur de 3 ou de 5, il faut rapprocher 3 boules unités de la barre transversale pour le premier cas, tandis que l'on doit rapprocher une boule supérieure de valeur 5 près de la barre transversale pour représenter 5. Ce système repose donc sur un même principe que l'abaque représenté à la figure 2.4.

En débutant par les poids les plus forts et non pas par les unités, le calcul sur boulier présente une différence importante avec notre manière moderne de calculer.

☞ La figure 2.6 illustre la somme de 458 et 227. Pour écrire 458, il faut déplacer 4 boules de poids 1 dans la colonne des centaines, une boule de poids 5 dans la 2^e colonne des dizaines, et une boule de poids 5 et trois boules de poids 1 (5 + 3) dans la colonne des unités. Pour additionner 227 à 458, on débute par le calcul des centaines. On devrait déplacer deux boules unités vers le haut, mais comme il n'y en pas assez, il faut calculer mentalement 4 + 2 = 6 et déplacer une boule de poids 5 vers le bas en ne gardant qu'une seule boule de poids 1 vers la barre transversale.

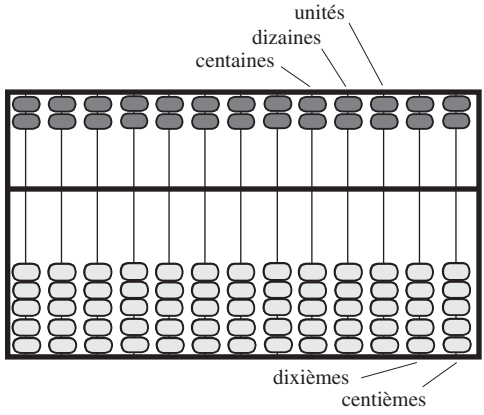


Fig. 2.5 Boulier chinois.

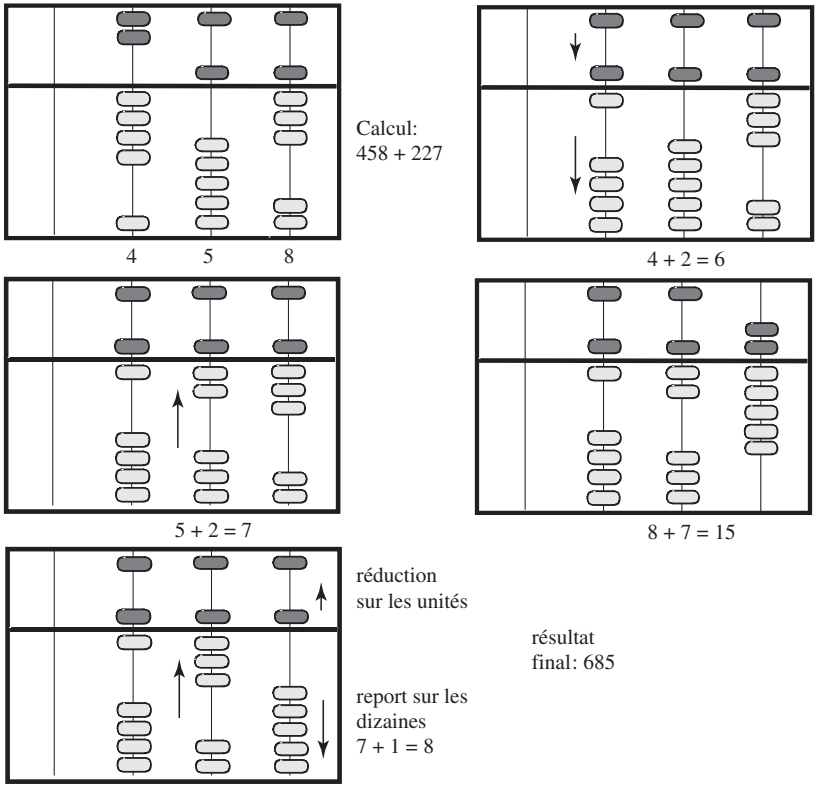


Fig. 2.6 Calcul avec un boulier.

Pour les dizaines, le calcul 5 + 2 est simple, puisqu'il suffit de monter deux boules unités vers le haut.
Pour les centaines, on doit ajouter 7 à 8. On descend donc une boule de poids 5 et on monte 2 boules de poids 1, conduisant à une valeur 15 affichée sur le boulier par les unités. Il faut alors procéder à des réductions afin qu'une colonne ne comporte pas de chiffre supérieur à 9. Pour les unités, on soustrait dix unités à 15, laissant 5 unités. On ajoute ensuite une dizaine supplémentaire à la colonne des dizaines, en montant une boule unité. Le résultat obtenu est 685.

On trouve aussi des bouliers plus simples, comportant quatre boules de poids 1 et une seule boule de poids 5, comme les bouliers japonais [3]. Ceux-ci ne permettent pas de représenter une valeur supérieure à 9 sur une même colonne, et les réductions s'effectuent de manière quelque peu différente qu'avec les bouliers chinois.

Les tables

Avec l'apparition de l'imprimerie, il devient possible de représenter dans des tables de calcul tous les résultats préalables.

Un très grand nombre de tables de multiplication, de logarithmes et autres furent publiées au XVIII^{e} siècle. En 1835, on rapporte qu'un mathématicien disposait dans sa bibliothèque de 140 volumes de tables [4]. C'était un énorme travail que d'établir ces tables à partir de calculs manuels, et les erreurs étaient fréquentes: on estime celles-ci à près de 3700 pour 40 volumes de tables, soit 40 erreurs par page. On comprend dès lors le besoin de machines capables d'établir ces tables sans erreur.

Les automates

Abaques, bouliers ou tables étaient des instruments de calcul manuel. Les instruments de calcul automatique vont permettre quant à eux de délivrer un résultat d'opération arithmétique sans intervention manuelle.

Avant de parler de calcul automatique, il convient de présenter les automates. Ceux-ci sont intéressants à plus d'un titre, car outre le fait qu'ils sont capables de calculer, ils sont aussi à l'origine des programmes aujourd'hui exécutés dans les ordinateurs.

Dans un automate, il faut enchaîner automatiquement plusieurs opérations, ce qui mène à la notion de séquence ou de programme. Le célèbre Héron d'Alexandrie [11, 20], au 1er siècle de notre ère, avait inventé un grand nombre d'automates, la plupart mus par l'eau, la vapeur ou des poids. Il utilisait déjà un cylindre en bois avec des chevilles pour animer des personnages, et construisit diverses clepsydres.

On rapporte aussi l'existence d'automates dès le III^{e} siècle de notre ère, réalisés pour le compte du Premier Empereur de Chine Ts'in Che-Houang Ti, bâtisseur de la Grande Muraille, et dont la tombe demeure gardée par les fameux 7000 soldats de

terre cuite [6]. Selon la légende, l'empereur trouva ces automates si parfaits qu'il en fit ouvrir un pour voir s'il ne s'agissait pas d'une supercherie.

Mais les automates les plus célébres sont mécaniques, comme le canard de Vaucanson (1709-1782), imitant parfaitement un volatile vivant [20], et ceux des Suisses Pierre et Henri-Louis Jaquet-Droz et de leur associé Jean-Frédéric Leschot, datant de 1770, et aujourd'hui conservés à Neuchâtel. Pierre Jaquet-Droz et son fils Henri-Louis étaient de célèbres horlogers de la Chaux-de-Fonds, artisans notamment de la célèbre pendulette du Berger que l'on peut admirer actuellement dans un palais de Madrid [22]. Les trois automates qu'ils concurent représentaient un écrivain (fig. 2.7a), un dessinateur (fig. 2.8) et une musicienne (fig. 2.7 à 2.10) [2].

L'Ecrivain, terminé en 1772, comporte le mécanisme le plus complexe des trois automates [2]. Le but était à l'époque de présenter au public un personnage mystérieux, en prenant soin de garder caché le mécanisme de la machine. Celui-ci est composé de deux parties distinctes (fig. 2.7b). La première est constituée d'un cylindre vertical sur lequel sont placées des cames qui donnent à la plume les informations nécessaires pour écrire une lettre, à raison d'une lettre par tour. Ce cylindre comporte 40 lettres et signes, et se déplace verticalement pour écrire l'un ou l'autre des caractè-

(a) (b)

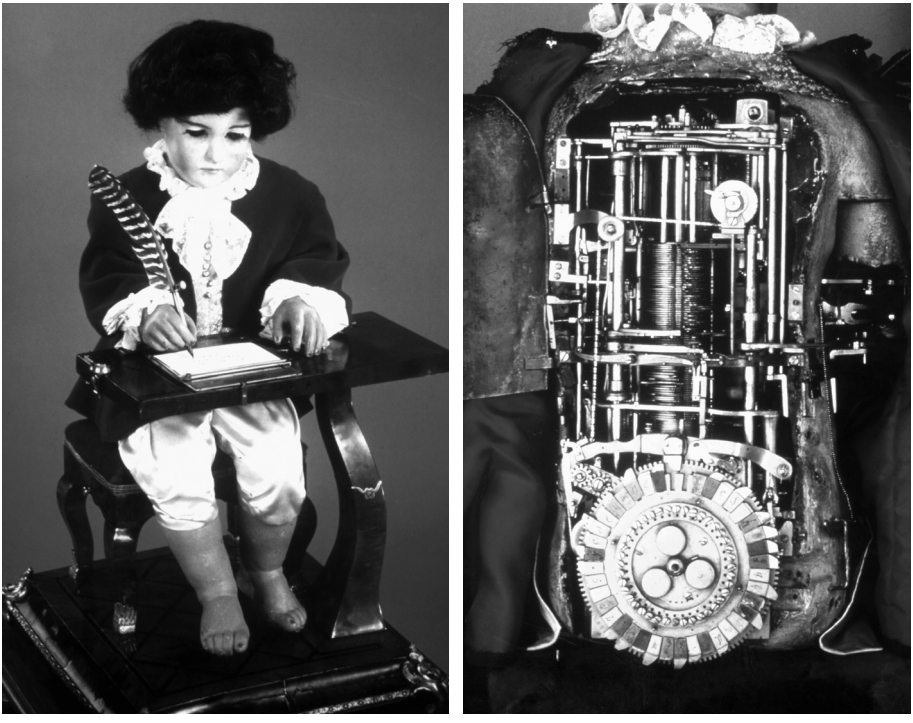


Fig. 2.7 L'Ecrivain (a) – Mécanisme de l'Ecrivain (b) (Musée d'art et d'histoire, Neuchâtel, Suisse).

res. La deuxième partie, placée en dessous, est un disque sur lequel on programme le texte que l'on veut faire écrire à l'automate. Ce mécanisme contrôle la hauteur du cylindre et les deux mécanismes travaillent en séquence. Pour composer un texte, on retire le disque de l'automate, afin de le configurer en choisissant les taquets commandant la hauteur du cylindre. Les 40 taquets sont placés en périphérie du disque. D'autres mécanismes permettent de déplacer la table pour écrire la lettre suivante, ainsi que de faire bouger les yeux et la tête de l'enfant.

Le Dessinateur, terminé en 1774, est un automate plus simple que l'Ecrivain, même si ses dessins semblent plus spectaculaires que le texte de l'écrivain [2] (fig. 2.8). Il peut réaliser quatre dessins différents, dont Cupidon et Louis XV. Le mécanisme comporte là aussi un cylindre vertical avec des cames, et chaque tour représente une portion du dessin (fig. 2.9). Un deuxième mécanisme permet ensuite de déplacer le cylindre afin d'exécuter une autre des 12 portions constituant le dessin. Les cames du cylindre doivent être changées pour exécuter un autre motif.

La Musicienne n'est quant à elle pas une simple boîte à musique avec un automate faisant semblant de jouer: c'est bien un véritable instrument (orgue) qu'emploie l'automate. Le cylindre est dans ce cas placé horizontalement, et commande les 5 doigts des deux mains (fig. 2.10).

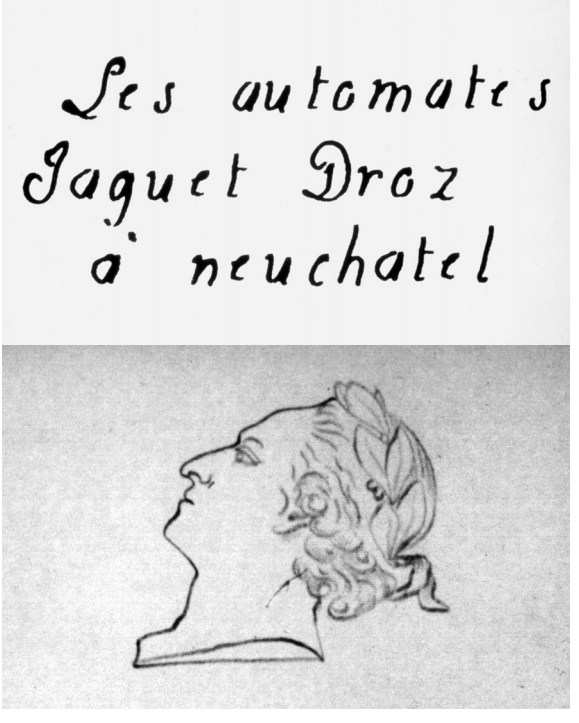


Fig. 2.8 Texte de l'Ecrivain et dessin du Dessinateur (Musée d'art et d'histoire, Neuchâtel, Suisse).

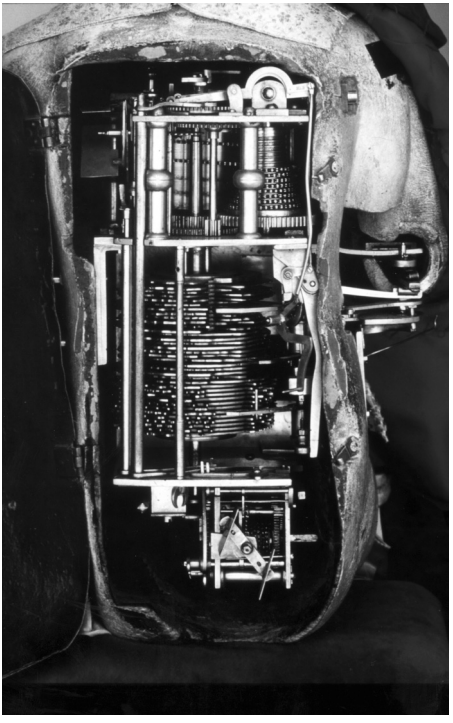


Fig. 2.9 Mécanisme du Dessinateur (Musée d'art et d'histoire, Neuchâtel, Suisse).

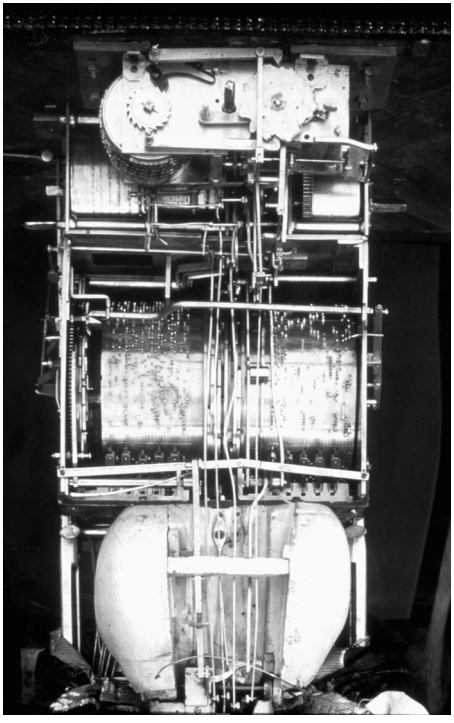


Fig. 2.10 Mécanisme de la Musicienne (Musée d'art et d'histoire, Neuchâtel, Suisse).

L'automate le plus simple qui apparaît à la même époque est sans doute la boîte à musique, toujours fabriquée de nos jours.

C'est en 1796 que le maître horloger genevois Antoine Favre (1734-1820) invente une machine mécanique dotée d'un programme enregistré sous forme de goupilles plantées dans un cylindre [5]. Lorsqu'on le fait tourner, il fait vibrer des lames d'acier, dont la séquence des sons produit une mélodie. La boîte à musique (fig. 2.11) présente donc une excellente analogie avec un programme binaire d'ordinateur, composé de «0» et de «1» (système de numération en base 2), puisqu'une goupille représente un «1» et l'absence de goupille un «0». Favre a inventé le peigne pour créer des sons, mais le principe du cylindre était connu. Inventé au XIV^{e} siècle, il permettait d'actionner les marteaux des cloches ou des carillons de pendules.

La boîte à musique est en quelque sorte un ordinateur mécanique rudimentaire.

La figure 2.12 représente le schéma d'une boîte à musique ainsi que de son modèle fonctionnel [18]. Le cylindre est une mémoire contenant des ordres codés en binaire;

entraîné par son moteur, il tourne comme un compteur incrémenté pour lire l'ordre suivant dans la mémoire. Ces ordres sont successivement, et toujours dans le même ordre, présentés à une unité de traitement représentée ici par le peigne. Il n'y a pas de possibilité de branchement, inutile pour une mélodie qui reste toujours la même. Cette machine est appelée *séquenceur*, car elle ne permet que de séquencer des ordres.

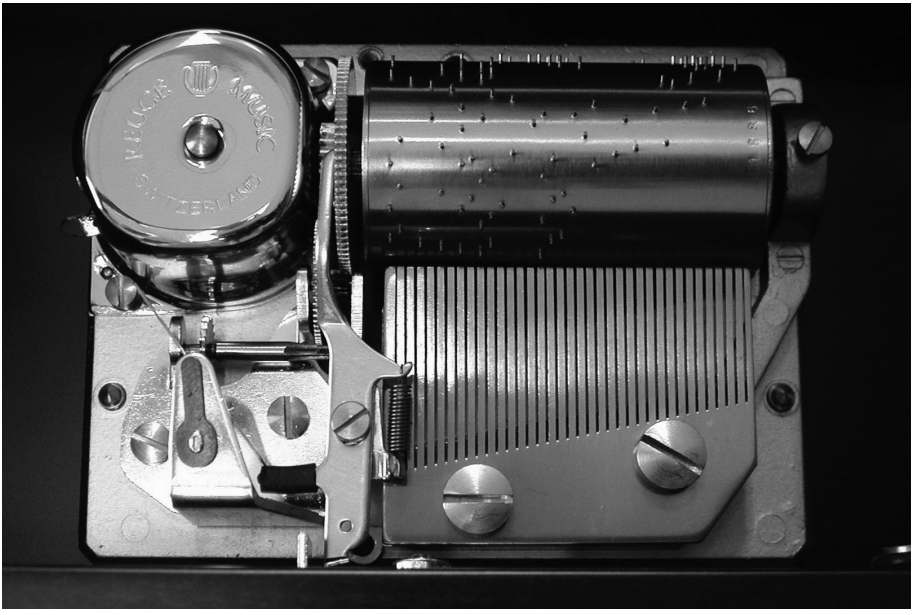


Fig. 2.11 Boîte à musique.

Le métier à tisser est d'un principe très similaire à celui de l'automate ou de la boîte à musique. Seule diffère l'unité de traitement.

Le principe de la commande demeure en effet le même pour ces différentes machines. Le cylindre à goupilles de la boîte à musique est analogue à la bande papier perforée ou aux cartes perforées utilisées par les métiers à tisser où un trou indique un «1» et une absence de trou un «0». Le premier métier à tisser utilisant des instructions sur papier perforé est inventé en 1725 par le tisserand B. Bouchon. Trois ans plus tard, Falcon imagine un système de commande similaire, mais à base de plaquettes de bois perforées. Il faut attendre 1749 pour que Vaucanson réalise un métier à tisser entièrement automatique, commandé par un cylindre analogue à ceux des boîtes à musique, et entraîné par un moteur hydraulique. Enfin, Joseph-Marie Jacquard (1752-1834) construit au début du XIX^{e} siècle les premiers métiers à tisser programmés par cartes perforées. Rappelons que les cartes et bandes perforées seront utilisées pour les ordinateurs jusqu'au début des années 1970.

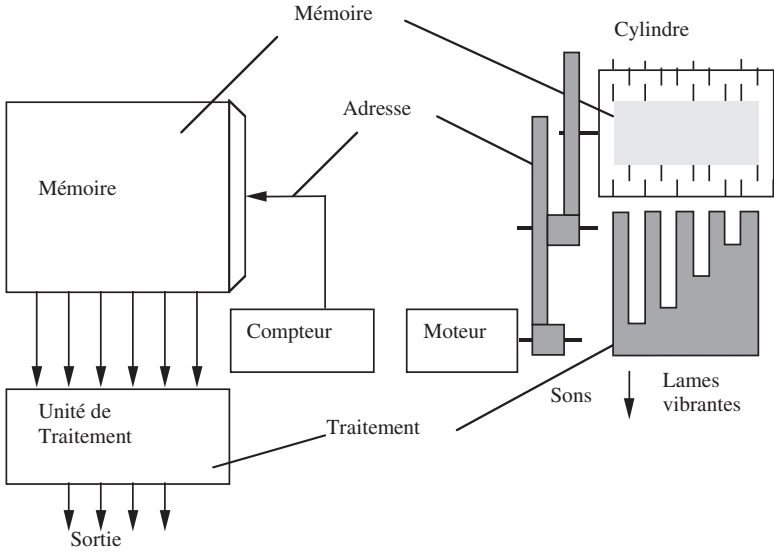


Fig. 2.12 Modèle d'une boîte à musique.

Calculatrices mécaniques

Les préalables à l'invention de la machine à calculer mécanique sont la numération de position et la connaissance du zéro. Ce n'est qu'au XVIII^e^ siècle que certaines de ces machines apparaissent, grâce au progrès réalisés en horlogerie et en mécanique théorique.

Bien que l'Eglise soutenait que le calcul ne pouvait relever que de l'esprit et était impossible à mécaniser [1], Wilhelm Schickard (1592-1635), professeur à Tübingen, mit au point la première machine à calculer mécanique, appelée *horloge à calculer*, capable de réaliser les quatre opérations (fig. 2.14). Les plans furent perdus et la machine détruite dans un incendie en 1624, mais on parvint néanmoins à la reconstituer, notamment à partir des lettres que Schickard envoyait fréquemment à Kepler et qui décrivaient l'appareil [4].

☞ Le principe d'une machine à calculer mécanique, illustré à la figure 2.13, est le suivant. La machine possède une roue mécanique dont la position représente un chiffre de 0 à 9, ainsi qu'une deuxième roue indiquant les dizaines, et ainsi de suite. La position de ces roues indique un nombre, et si l'on veut ajouter un autre nombre, on fait tourner la roue des unités de la valeur du nombre à ajouter. Ainsi, si la valeur de départ est égale à 15 et que l'on ajoute 3, on fera tourner la roue des unités de 3 positions, et la machine indiquera 18.

Le principal problème de toute calculatrice est le report des unités sur les dizaines ou celui des dizaines sur les centaines. Pour le réaliser, une roue intermédiaire fut placée entre les unités et les dizaines, chaque tour de roue des unités entraînant l'avance

d'une dent sur la roue des dizaines (fig. 2.13). Une telle chaîne de reports exigeait toutefois une force qui pouvait casser les dents de la roue des unités.

Ces roues à dix dents utilisent un système de numération décimal, permettant à la fois de réduire le nombre de roues par rapport à un système binaire, mais aussi d'afficher aisément les résultats sans devoir les convertir en décimal.

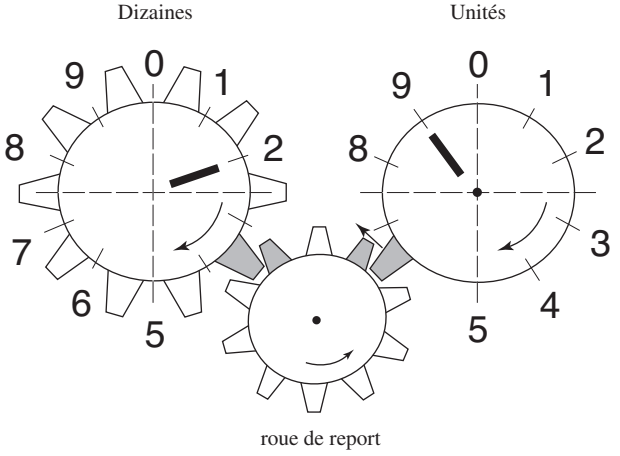


Fig. 2.13 Mécanisme de report des unités sur les dizaines.

Fig. 2.14 Machine de Schickard (Musée de Paderborn, Allemagne).

A l'âge de 19 ans, Blaise Pascal (1623-1662) [23] invente une calculatrice mécanique, la Pascaline, sans connaissance préalable des travaux de Schickard.

Cette machine, produite à quelques dizaines d'exemplaires, peut additionner et soustraire; elle comporte des roues représentant les chiffres 0 à 9 avec un report sur la roue d'ordre supérieur [3] (fig. 2.15). Pascal sait qu'une propagation de reports sur toutes les roues ne peut fonctionner correctement. Il réalise alors un mécanisme complètement différent de celui de Schickard: la roue des unités entraîne lors du passage de 0 à 9 un levier qui tombe par gravitation sur les pointes de la roue des dizaines, laquelle avance alors d'une position (fig. 2.16) [4]. Ceci implique que les roues ne peuvent tourner que dans un sens, celui de l'addition. Pour soustraire, Pascal ajoute des nombres compléments à 9. Mais lorsqu'un grand nombre de reports successifs survient, la machine se bloque. D'autres parviennent à éviter ce problème en construisant des machines sur lesquelles les reports sont propagés manuellement, suivant les indications de la machine. C'est par exemple le cas de la machine à additionner de Samuel Morland, construite en 1666 et décrite dans son livre en 1673 [4].

Fig. 2.15 Machine de Pascal, La Pascaline (Musée de Paderborn, Allemagne).

Gottfried Wilhelm Leibnitz (1646-1716) s'intéresse aux travaux de Pascal et réalise en 1673 une machine qui additionne et multiplie par additions et décalages successifs, sur le principe de la multiplication égyptienne [4].

L'invention de Leibnitz consiste en un cylindre doté de dents de longueurs différentes et capable d'entraîner une roue dentée de 0 à 9 pas selon sa position le long du cylindre (fig. 2.17). La machine comporte au total 8 de ces cylindres. Après avoir

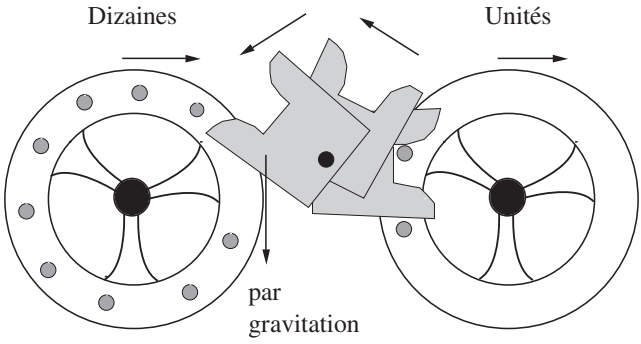


Fig. 2.16 Mécanisme de report de la Pascaline.

introduit le nombre à multiplier en positionnant correctement les roues dentées le long des cylindres, on tourne le cylindre d'un nombre de fois égal à la valeur du multiplicateur. Le report d'un digit sur le suivant est assuré par un mécanisme muni de leviers et de roues, qui doit forcer un pas de la roue dentée d'ordre supérieur. La machine demande cependant une assistance manuelle pour une chaîne de reports.

En 1694, Leibnitz construit une machine à calculer en base 2, beaucoup plus complexe que celle de Pascal, mais qui ne fonctionnera jamais vraiment (fig. 2.18). Leibnitz comprit cependant que cette base la plus simple pouvait être utilisée avec profit pour une mécanisation ou une automatisation des calculs.

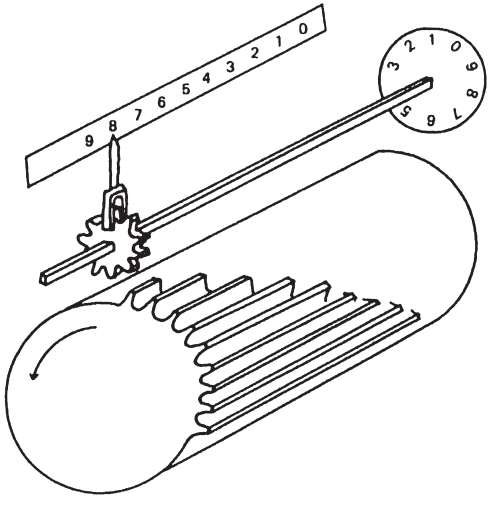


Fig. 2.17 Cylindre de Leibnitz
(M. R. Williams, «History of Computer Technology», IEEE Computer Society, 1979).

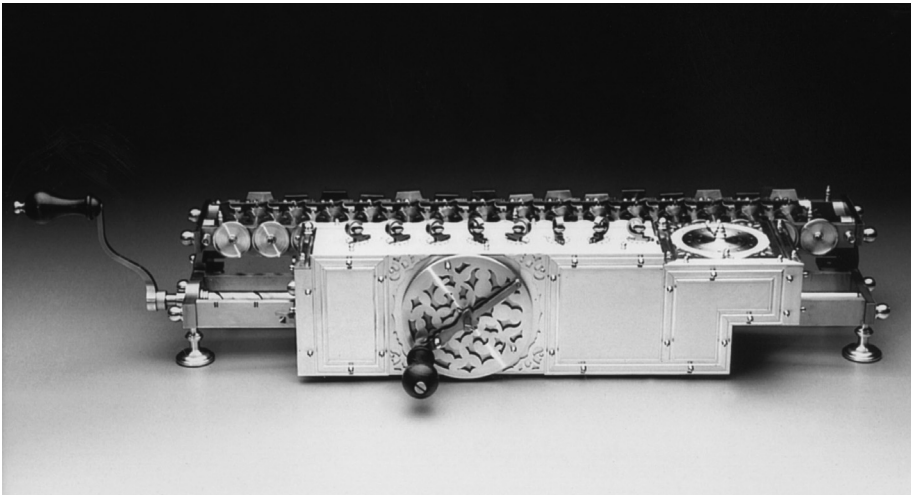


Fig. 2.18 Machine de Leibnitz (Musée de Paderborn)

Machines à différences

En 1821, Charles Babbage présente à la Royal Astronomical Society une machine dite «à différences», capable de calculer les tables nécessaires à la détermination de la position des planètes.

Babbage (1791-1871) avait une passion pour la précision et déplorait les innombrables erreurs que l'on trouvait dans les tables mathématiques de l'époque [21]. Ceci le poussa à réaliser son propre appareil de calcul. Une fois sa machine «à différence» mise au point pour les calculs astronomiques, il refléchit à la réalisation d'une machine programmable, dont le principe deviendra celui d'un ordinateur, et que l'on retrouvera sous la forme de la machine analytique.

Babbage n'est sans doute pas l'inventeur de la machine à différences, car il semble que le principe ait été inventé 30 ans plus tôt, en 1786, par un ingénieur allemand du nom de J.-H. Müller. Le principe de cette machine dite «à différences» est d'évaluer la valeur d'un polynôme, par exemple $F(x) = 2x + 3$ pour toutes les valeurs de x, sans avoir à réaliser la multiplication $2 \times x$ (fig. 2.19).

x	1	2	3	4	5	6
F(x)	5	7	9	11	13	15
différence	2	2	2	2	2	

Fig. 2.19 Table de $F(x) = 2x + 3$.

☞ Si l'on calcule F(x) pour chaque valeur de x, on a pour x = 1, F(1) = 2 × 1 + 3 = 5, pour x = 2, F(2)=2 × 2 + 3 = 7, pour x = 3, F(3) = 2 × 3 + 3 = 9, et ainsi de suite. Si l'on considère la différence entre ces résultats successifs, soit 5, 7 et 9, on constate que celle-ci est toujours constante, et égale à 2. Pour évaluer la valeur de polynôme F(x) sans avoir à effectuer la multiplication 2 × X, il suffit donc de calculer la valeur de F pour x = 1, et la suite des résultats F(x) jusqu'à x = X. La première valeur F(1) = 5 ne nécessite pas de multiplication (car multiplier par 1 équivaut à l'identité); pour les résultats suivants, on ajoute 2 pour obtenir les valeurs suivantes de F(2), F(3), F(4), soit 7, 9, 13, etc. jusqu'à la valeur cherchée de F(x).
Si le polynôme est de degré plus élevé, tel $F(x) = x^2 + 2x + 3$, il s'agit de former deux différences afin d'obtenir une constante dans la deuxième différence, comme l'illustre la table de la figure 2.20.

x	1	2	3	4	5	6
F(x)	6	11	18	27	38	51
1re différence	5	7	9	11	13	
2e différence	2	2	2	2	2	

Fig. 2.20 Table de $F(x) = x^2 + 2x + 3$.

Dans ce cas, la machine ne peut calculer directement F(x). Elle doit d'abord déterminer la première différence, en ajoutant la constante 2, puis ajouter cette différence pour calculer les valeurs du polynôme. Comme l'illustre la figure 2.21, ceci nécessite deux étages. Les quatre étapes de calcul que demande la détermination de F(4) y sont par ailleurs illustrées: pour x = 1, on doit déterminer les premières valeurs F(1) = 6, la première différence, égale à 5, et la deuxième différence, égale à 2. Lors de la deuxième étape, on calcule la nouvelle première différence, égale à l'ancienne première différence + 2 (soit 7 = 5 + 2), et F(2) = F(1) + ancienne première différence (soit 11 = 6 + 5). Pour la troisième étape, on a alors la nouvelle première différence égale à 9 (7 + 2), et F(3) = F(2) + ancienne première différence, soit 18 (11 + 7). Pour la quatrième étape, on a une nouvelle première différence de 11 (9 + 2) et F(4) = 27 (18 + 9).

La machine à différences de Babbage reposait sur ce principe, et pouvait aller jusqu'à la sixième différence pour des nombres de 18 digits. Il y avait donc 6 étages de roues permettant de calculer les différences, ainsi qu'un mécanisme d'addition avec report pour évaluer les valeurs du polynôme. Deux versions de cette machine furent imaginées, mais aucune ne fut totalement terminée. Il faudra attendre 1991, soit 200 ans après la naissance de son inventeur, pour que la machine à différences de Babbage voit le jour au Science Museum de Londres. Elle pèse près de trois tonnes et comporte 4000 pièces.

En 1843, le Suédois Charles Scheutz et son fils furent les premiers à construire une machine à 5 digits ayant jusqu'à trois différences, grâce aux descriptions de la machine de Babbage [4] et à un support financier du gouvernement. Dix ans plus tard, ils mirent au point une deuxième machine fonctionnelle avec 15 digits, capable d'aller jusqu'à la quatrième différence.

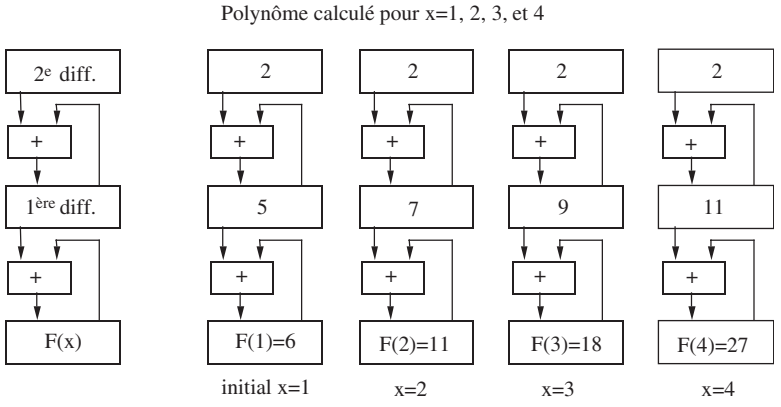


Fig. 2.21 Machine à différences de Charles Babbage, $F(x) = x^2 + 2x + 3$.

Machines mécaniques avec programmes

Babbage parvint à dissocier les fonctions d'une roue, soit le stockage d'un digit, et le calcul. Il proposa en 1834 une autre machine, entièrement mécanique et programmable, capable d'exécuter n'importe quelle opération arithmétique, et de poursuivre les calculs en fonction des résultats intermédiaires: le «moteur analytique» [15].

Les fonctions de cette machine étaient clairement distinctes, et se répartissaient entre fonctions de mémoire («store»), fonctions de calcul («mill», le moulin) et l'unité de commande («control barrel», soit le séquenceur des opérations de calcul). Une telle machine [4] comportait donc des branchements conditionnels, et offrait la possibilité de décider en fonction d'un résultat intermédiaire quelle partie du programme devait être exécutée. La mémoire de données de l'appareil comportait environ 40 registres de 40 digits décimaux. Les nombres traités comportaient autant de digits car la précision désirée pour les tables mathématiques de l'époque était grande, et une machine utilisant des nombres en «virgule flottante» (comme $1{,}275864 \times 10^{22}$) était beaucoup trop compliquée à réaliser en mécanique. Mieux valait donc porter le nombre de digits à 40.

L'unité de calcul fonctionnait grâce à des roues à 10 positions représentant les chiffres de 0 à 9, et utilisait donc le système décimal. Le système binaire ne sera choisi que lorsque les machines deviendront électromécaniques ou électroniques.

Pour une machine telle que celle-ci, entièrement mécanique, et dont le fonctionnement repose sur des roues dentées, le système décimal est assurément le plus approprié. Un nombre de 40 digits est ainsi determiné par la position de 40 roues dentées.

☞ La figure 2.22 représente une roue dentée à dix positions. Dans l'une d'elles, la valeur du chiffre est «0». Si la roue est tournée de 2 ou 3 positions, sa valeur devient alors égale à 2 ou à 3. Cette roue est entraînée par une autre (non représentée), et l'axe est monté de telle

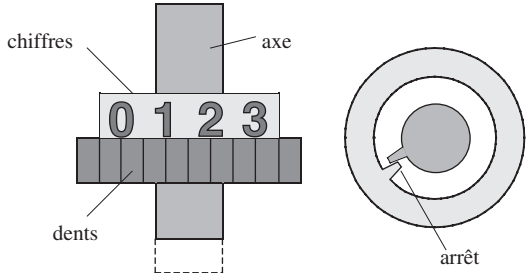


Fig. 2.22 Roue dentée de la machine analytique de Babbage.

sorte que les deux ergots servant d'arrêt soient à la même hauteur. Pour lire la valeur de l'une des roues, on procède comme suit: on tourne cette roue jusqu'à ce que les ergots se touchent, et indiquent la valeur «0». Le nombre de positions dont la roue vient de tourner est alors égal à la valeur cherchée, ou à transmettre ailleurs dans la machine [15].

Le problème de cette lecture est qu'elle est destructive, puisque la roue est alors en position «0». Ceci n'est pas le cas des équivalents électroniques, les registres, dont les lectures ne sont pas destructives. Pour résoudre ce problème, Babbage double le nombre de roues associées à un chiffre donné afin d'éviter de perdre celui qu'on lit.

Comment réaliser une addition avec ce type de roue?

☞ La figure 2.23 illustre l'exemple de 7 + 2. Ici, la roue de droite affiche 2 et celle de gauche 7. On fait alors tourner cette dernière de 7 positions, jusqu'à ce qu'elle soit arrêtée à la position «0». Ces 7 positions sont transmises par l'intermédiaire de deux petites roues à la roue de droite, qui affiche alors 9.

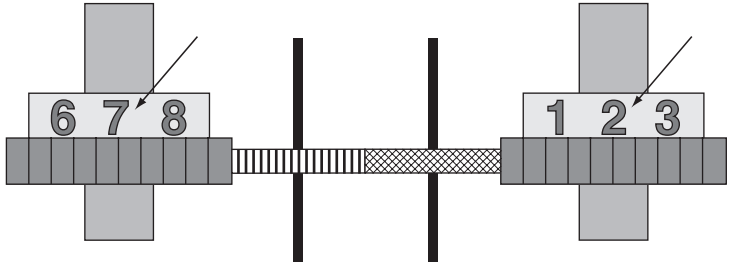


Fig. 2.23 Addition avec la machine analytique de Babbage.

Babbage résolut le problème des reports en ordonnant les différentes roues associées aux unités, dizaines et centaines d'un même nombre les unes au-dessus des autres. Les unités étant placées en bas, il fallait, en cas d'un report, transmettre l'indication à la roue placée directement au-dessus. Ceci était assuré par des roues spéciales capables de détecter le passage du 9 au 0 et de faire tourner la roue du dessus d'une position.

Si le report devait être transmis en chaîne, c'est-à-dire des dizaines aux centaines, des centaines aux milliers, et ainsi de suite, le processus pouvait prendre beaucoup de temps. Babbage imagina alors une anticipation du report permettant de le transmettre simultanément aux roues qui devaient en tenir compte [15]. Cette technique a ensuite été utilisée dans les calculateurs électroniques sous la forme de portes de transmission: elle est aujourd'hui connue sous le nom de «Manchester».

L'unité de calcul de la machine analytique de Babbage est proche de celles mises au point par Pascal et Leibnitz. Mais la grande nouveauté de l'appareil est le séquenceur des opérations de calcul (control barrel).

Le séquenceur est constitué d'un cylindre – peut-être inspiré de celui des boîtes à musique – poussant des barres afin de réaliser les différentes étapes d'une opération de calcul. Il peut à tout moment indiquer l'étape suivante, et sa rotation est commandée par un compteur indiquant à quel stade de l'opération se trouve le processus (fig. 2.12). L'analogie avec un séquenceur de microprocesseur doté d'un compteur de microprogramme et d'une mémoire de microcode est donc parfaite, car ce cylindre permet ainsi de séquencer les étapes d'une opération de calcul ou d'une instruction. Babbage estimait à 3 secondes le temps nécessaire pour effectuer une addition.

Il restait cependant à enchaîner les instructions les unes aux autres afin d'obtenir un programme complet. Pour cela, Babbage s'inspira du métier à tisser de Jacquard, et inscrivit chacune des instructions du programme sur des cartes perforées lisibles par la machine. Un deuxième lecteur permet l'introduction des données. Les boucles et les branchements sont possibles, une instruction pouvant indiquer un retour ou un saut à quelques instructions en arrière ou en avant, et même de ne pas exécuter les quelques instructions suivantes.

La machine présente donc deux niveaux: l'un permet de gérer le programme, l'autre d'exécuter l'ensemble des étapes nécessaires à l'exécution d'une même instruction.

La figure 2.24 représente un modèle abstrait de la machine analytique de Babbage, et illustre son fonctionnement. Le programme est constitué d'une suite de cartes perforées, et chacune de ces instructions comporte une zone indiquant la suite des étapes nécessaires à l'exécution de l'instruction. En informatique ces étapes sont appelées «micro-instructions», ou microprogramme spécialisé dans l'exécution de l'instruction correspondante. Dans la machine de Babbage, elles sont indiquées sur le cylindre, qui se déplace de manière à sélectionner celles correspondant à l'instruction. Ces micro-instructions commandent l'unité de calcul afin de réaliser la bonne opération. Leur séquence est définie par un pointeur P, qui est soit incrémenté (+1), soit chargé par une indication de branchement contenue dans la micro-instruction elle-même, ce qui permet de revenir à une étape antérieure (boucle) ou de sauter à une étape ultérieure (branchement). Ce processus est par ailleurs soumis aux conditions de branchement (type résultat intermédiaire) fournies par l'unité de calcul. Babbage s'est lui-même étonné de la puissance de la machine qu'il avait imaginée.

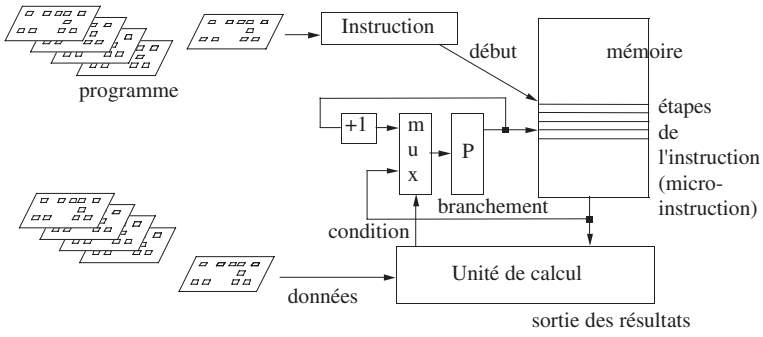


Fig. 2.24 Modèle de la machine analytique de Babbage.

Les programmes de cette machine ont été réalisés par la première programmeuse de l'histoire de l'informatique, la comtesse de Lovelace, Augusta Ada Byron (1815-1852), fille du poète anglais Lord Byron.

Cette femme contribua à la réalisation de la machine de Babbage, en élaborant notamment une description qui permit d'en comprendre les concepts. Elle inventa par ailleurs la technique des sous-programmes appelés plusieurs fois depuis un programme principal [19]. C'est en son hommage qu'un langage de programmation moderne s'appelle Ada.

Entièrement mécanique, la machine de Babbage ne fut jamais terminée. L'un de ses éléments, le moulin, ne fut construit qu'en 1906 par le propre fils de l'inventeur.

Les travaux de Babbage ont été, à leur époque, complètement ignorés et ils demeurent les seuls de leur temps sur de tels appareils. La structure de la machine analytique de Babbage est assez proche de celles de certains ordinateurs électroniques, notamment ceux qui seront microprogrammés. En 1939, Howard Aiken, de l'Université de Harvard, s'inspirera de ces travaux vieux de plus d'un siècle pour construire son ordinateur électromécanique, le Mark I.

Commercialisation de calculatrices mécaniques

La commercialisation des premières calculatrices mécaniques, alors inspirées des cylindres de Leibnitz, est lancée en 1822 par l'industriel français Charles-Xavier Thomas. En 1875 apparaissent les machines américaines (Baldwin) et russes (Odhner).

Le problème majeur de ces premières calculatrices mécaniques est leur lenteur par rapport au calcul manuel, l'introduction des données étant longue et non exempte d'erreurs.

La génération suivante de machines à calculer mécaniques est dotée d'un clavier numérique tel des machines à écrire. Ces nouvelles calculatrices datent de 1850

(Remington, Hansen), mais il faut attendre 1887 pour que leur commercialisation débute à Chicago. Afin d'être lu, le résultat doit être imprimé; les machines Burrough avec imprimante connaissent un grand succès dès leur apparition, en 1885. Simultanément, vers 1879, apparaissent des machines capables d'effectuer des multiplications et divisions de manière directe, sans passer par une suite d'additions et de décalages (type multiplication égyptienne). Le Français Léon Bollée, inventeur de la première voiture à essence et des 24 Heures du Mans, propose en 1888 une machine à multiplication directe dotée d'une table de multiplication.

Selon la Constitution des Etats-Unis, un recensement de la population doit être fait tous les dix ans. Comme 7 années avaient été nécessaires au dépouillement de celui de 1880, un concours est ouvert pour proposer une solution mécanique au recensement de 1890. Hermann Hollerith (1869-1929) le gagne grâce à son totalisateur électrique. L'appareil est programmé par des cartes perforées comportant tous les renseignements relatifs à un même individu, et peut compter et trier les informations selon les critères désirés. Il est doté de tiges mobiles, qui, au passage de la carte, sont soit stoppées (absence de trou), soit pénétrantes (présence de trou). Lorsqu'il y a pénétration de la tige, un contact électrique est établi et un compteur réalise l'opération + 1. En 6 semaines, Hollerith annonce les résultats du recensement: 62 622 250 personnes constituent la population des USA de 1890 [3]. En 1911, il participe à la création de la Tabulating Machine Company, qui deviendra IBM en 1924.

En 1910, les machines à calculer mécaniques sont dotées d'un moteur électrique. Trente ans plus tard, les premières machines à relais électromécaniques sont commercialisées. L'évolution des machines à calculer va dès lors se poursuivre dans trois directions distinctes. L'une ménera aux premiers ordinateurs, dédiés exclusivement au calcul. Une autre conduira aux calculatrices portables mécaniques, telles que la Curta inventée en 1948 par Herzstark, au Liechtenstein, et qui pesait 230 grammes (fig. 3.9b). Une troisième voie débouchera sur les calculatrices électroniques, mises sur le marché au début des années 1970 par Texas Instruments et Hewlett-Packard, et qui provoqueront la disparition de la Curta et de la règle à calcul.

L'algèbre de Boole

Les calculatrices mécaniques utilisaient à la fois la base 2 et la base 10. Les ordinateurs n'utilisent quant à eux que la base 2.

Georges Boole (1815-1864) fut le premier, en 1854, à faire de la logique une branche des mathématiques [3] et à démontrer la théorie de la logique des propositions «vrai» et «faux». Il inventa du même coup les opérateurs logiques comme le ET, le OU et le NON [7].

Dès 1679, Leibnitz avait pressenti que la réalisation de machines à calculer serait plus aisée en employant une logique de type booléenne ou binaire. Cette intuition se vérifiera abondamment par la suite avec les machines électroniques.

Les principaux concepts

Vers le milieu du XIX^{e} siècle, la structure générale d'une machine de traitement de l'information est connue, notamment grâce aux remarquables descriptions de Ada Lovelace. Cette structure consiste en une unité de calcul (le moulin), une mémoire de stockage des données, une unité de commande capable de séquencer les opérations en lisant un programme (inscrit sur cartes perforées) et deux organes d'entrée et de sortie [3]. La logique binaire décrite par Leibnitz et Boole permet une grande simplification de la partie unité de calcul.

L'absence de toute électronique rend toutefois impossible le développement de machines plus ambitieuses. Il faudra attendre près d'un siècle et que la technologie progresse pour permettre la réalisation de structures de machines semblables à celles décrites par Babbage et Ada Lovelace. Les premières unités de commande sont réalisées avec des cylindres analogues à ceux des boîtes à musique ou grâce à des lecteurs de cartes perforées de métiers à tisser, soit des séquenceurs. Quant aux unités de calcul, elles sont mises au point avec des rouages fonctionnant à la manière d'un boulier. L'association d'un cylindre et d'un boulier a donc été le premier stade d'une évolution conduisant aux architectures des ordinateurs modernes. Ne manquait alors qu'un concept essentiel, auquel Babbage a apporté les premières solutions: les branchements conditionnels, qui n'apparaîtront que 100 ans plus tard avec les ordinateurs électromécaniques et électroniques.

Recommandons pour terminer la visite du musée de Paderborn [8], en Allemagne, qui présente des copies des premières machines à calculer, ainsi que des modèles des premiers ordinateurs.

Remerciements

Les photos des automates Jaquet-Droz ont été réalisées par le Musée d'art et d'histoire de la Ville de Neuchâtel, en Suisse. Les photos de la machine de Schickard, de la Pascaline et de la machine de Leibnitz sont celles du musée Heinz Nixdorf Museum, à Paderborn.

Références

[1] Georges Ifrah, *Histoire universelle des chiffres*, Bouquins, Robert Laffont, 1994.

[2] A. Chappuis, E. Droz, *Automata*; A. Chappuis, E. Droz, *Les automates des Jaquet-Droz*, Musée d'histoire de la Ville de Neuchâtel.

[3] Alain Taurisson, *Du boulier à l'informatique*, La Cité des Sciences et de l'Industrie, Explora, Presses Pocket, 1991.

[4] M. R. Williams, *History of Computing Technology*, IEEE Computer Society Press, Los Alamitos, CA, second edition, 1997.

[5] F. Rausser, D. Bonhôte, F. Baud, *Au temps des boîtes à musique*, Editions Mondo, 1972.

[6] Jean Lévi, *Le Grand Empereur et ses automates*, Albin Michel, 1985, Poche 6239.

[7] Gaston Casanova, *L'Algèbre de Boole*, Coll. Que sais-je? PUF n° 1246, 1972.

[8] Heinz Nixdorf Museum, Fürstenallee 7, D-33102 Paderborn (http://www.hnf.de)

[9] Philippe Breton, *Une histoire de l'informatique*, Coll. Points S65, Seuil, 1990

[10] H. D. Huskey, V. R. Huskey, *Chronology of Computing Devices*, IEEE Trans. on Computers, Vol. C-35, N° 12, December 1976, pp. 1190-1199.

[11] P. Devaux, *Automates, Automatisme, Automation*, Coll. Que sais-je? PUF n° 29, 6e édition, 1967.

[12] John P. Hayes, *Computer Architecture and Organization*, McGraw-Hill Book Company, 1978.

[13] André Rochat, «Histoire de la bureautique et de l'informatique», *Flash Informatique*, EPFL Informatique, 15 septembre 1998, pp. 17-21.

[14] M. Ascher, *Mathématiques venues d'ailleurs*, Seuil/sciences ouverte, 1998

[15] A. G. Bromley, «Charles Babbage's Analytical Engine, 1838», IEEE Annals of the History of Computing, Vol. 20, n° 4, 1998, pp. 29-45.

[16] «L'univers des nombres», *La Recherche* N° 322, hors série, 2 août 1999

[17] Jean-Yves Pouilloux, *Montaigne, Que sais-je?*, 23/Découvertes Gallimard/Littérature, 1987

[18] J-F. Perotto, C. Piguet, «Qu'est-ce qu'un processeur?», SSC, 52e Congrès, 1977, pp. 305-8.

[19] B. A. Toole, «Ada Byron, Lady Lovelace, An Analyst and Metaphysician», IEEE Annals of the History of Computing, vol. 18, n° 3, 1996, pp. 4-12.

[20] Bruno Jacomy, *Une Histoire des techniques*, Coll. Points S67, Seuil, 1990.

[21] C. Piguet, *Babbage, l'inventeur de l'ordinateur*, FTFC 2001, Paris, 30-31 mai, 1er juin 2001.

[22] André Tissot, «Le voyage de Pierre Jaquet-Droz à la cour du roi d'Espagne», *Cahiers de l'Institut neuchâteloise*, A La Baconnière, 1982.

[23] Jacques Attali, *Blaise Pascal ou le génie français*, Fayard, 2000.

CHAPITRE 3

HISTOIRE DES MACHINES À CALCULER ÉLECTRONIQUES

«Scientifiques! dotez-vous d'un génial second cerveau: le HP-35.»
Une publicité, 1975

Introduction

Les machines à calculer électroniques sont aujourd'hui devenues omniprésentes. Il n'aura fallu qu'une dizaine d'années pour qu'elles remplacent totalement les machines mécaniques et électromécaniques.

En 1950, les machines à calculer mécaniques et électromécaniques sont très répandues dans tous les domaines de la vie économique [1] [4] [12]. On trouve d'imposantes caisses enregistreuses équipées de systèmes à palettes dans la plupart des magasins et restaurants, ainsi que des machines à calculer capables d'effectuer les additions et soustractions requises en bureautique pour les décomptes et la comptabilité (fig. 3.1). Le mécanisme de calcul de ces machines est généralement couplé en sortie à un système d'impression qui permet de reporter sur papier l'ensemble des nombres tapés et des résultats obtenus. Des travaux plus exigeants comme le calcul scientifique et technique poussent parallèlement au développement de machines plus performantes. L'aboutissement de cette évolution est notamment illustré par le modèle SRW de la marque Friden, qui offre diverses fonctionnalités comme les quatre opérations, la disponibilité de plusieurs registres et la fonctionnalité avancée du calcul de la racine carrée [9].

(a) (b) (c)

Fig. 3.1 Machines à calculer mécaniques:
(a) Anker env.1920 (b) Odhner (env. 1945) (c) Monroe (env. 1950) (photos André Rochat).

Malgré le large succès commercial remporté par les calculatrices mécaniques et électromécaniques, et aussi le bon niveau de fonctionnalité atteint dans leur développement, ces machines se heurtent à certaines limites liées aux contingences du fonctionnement mécanique. Les caractéristiques de masse, de frottement et de résistance des éléments en mouvement ne permettent de construire que des instruments lents, bruyants, soumis à l'usure, et limités dans leur capacité de développement et de miniaturisation.

Le remplacement de la mécanique par l'électronique laisse entrevoir des solutions à tous ces problèmes. L'histoire des calculatrices électroniques commence alors à être intimement liée à celle de l'électronique.

L'électronique repose successivement sur les tubes à vide, les transistors et les circuits intégrés.

Les tubes électroniques à vide ne sont pas suffisamment compacts pour offrir une solution. Le transistor, inventé en 1947, offre une première perspective d'une électronique condensée et peu demandeuse d'énergie. Il faut cependant attendre l'invention du circuit intégré en 1959 pour disposer des fondements technologiques nécessaires au développement d'une calculatrice électronique compacte. La première calculatrice électronique, nommée Anita, est construite en 1962. La commercialisation des calculatrices électroniques débutera deux ans plus tard, et fera pendant près de 10 ans l'objet d'une lutte serrée entre de nombreuses entreprises tant aux Etats-Unis qu'au Japon. Plusieurs sociétés vivent l'arrivée des machines à calculer électroniques avec intérêt et jouent un rôle important dans leur développement.

Casio

Comme tant d'autres innovations, les machines à calculer naissent au carrefour de la technique et du marché. L'histoire de Casio est à cet égard exemplaire.

D'origine modeste, le Japonais Tadao Kashio (1917-1993) [8] dirige un atelier de mécanique lorsque l'empereur annonce la fin de la Seconde Guerre mondiale. Il lui faut alors rapidement substituer à la production de guerre une manufacture de biens de consommation courante. L'entreprise passe alors de la fabrication de munitions à celle des casseroles, puis des dynamos pour bicyclettes, ceci en collaboration avec six autres PME. Mais l'entreprise Kashio Seisakujo, fondée en 1946, souhaite lancer d'autres produits. Elle rencontre un premier succès avec la bague porte-cigarette, un gadget offrant la possibilité de fumer une cigarette dans sa totalité, ou de fumer pendant le travail. Le produit se vend bien mais le marché s'épuise vite. A la recherche de nouveaux produits, Kashio est impressionné par une calculatrice électromécanique exposée dans un grand magasin de Tokyo (fig. 3.2).

Il n'existe alors pas de fabricants de calculatrices au Japon, et les machines importées sont très chères. Décelant là un marché potentiel important et profitable à long

terme, Kashio pousse son associé Toshio à se consacrer entièrement au développement d'une calculatrice électrique. L'entreprise part de rien, tout est à inventer. Une première calculatrice à base de solénoïdes est alors longuement développée et finalement réalisée en 1954. Ses capacités s'avèrent toutefois nettement inférieures à celles des machines électromécaniques traditionnelles. Peu découragés par ce semi-échec et plutôt confortés dans leur projet par les difficultés rencontrées dans ce domaine par d'autres entreprises plus importantes, Kashio et Toshio se lancent dans le développement d'une calculatrice entièrement électrique à base de relais. La machine occupe près d'un demi-mètre cube et pèse quelque 120 kg. Ces grandes dimensions excèdent les limites autorisées pour le transport par avion. Présentée lors d'une exposition à Sapporo, elle doit être démontée avant son transport, puis ré-assemblée dans la hâte à son arrivée. Malgré un fonctionnement partiel le jour de la présentation – les fonctions de multiplication et de division ne fonctionnent pas – et un succès mitigé, un client se montre intéressé par la machine et procure suffisamment de moyens pour poursuivre le développement de nouvelles calculatrices.

(a)

(b)

Fig. 3.2 De l'enthousiasme pour une calculatrice électrique (a) à la fabrication des premières calculatrices à relais (b) (Casio Computer Co.).

En 1957, Tadao Kashio fonde Casio Computer, Co. Ltd. L'entreprise compte 20 employés et projette de développer et produire des calculatrices à relais. Cinq mois plus tard, la Casio 14-A est lancée. Cette calculatrice de table comprend 342 relais et opère sur des nombres avec une précision à 14 chiffres. Fabriquée à l'origine sur commande, elle rencontrera un succès avec 19 machines commandées. En 1959 apparaît le modèle 14-B, à vocation technique et capable de calculer la racine carrée. Suivent d'autres machines, comme le modèle scientifique Casio 301, puis la Casio programmable AL-1. En 1962, le nombre d'employés est passé à 300 et l'entreprise prospère.

Mais l'arrivée du transistor met la technologie des relais adoptée par Casio en péril. Kashio se sent particulièrement menacé lorsqu'en juillet 1964, Sharp annonce le développement d'une calculatrice entièrement électronique. Plus d'une vingtaine de compagnies dans le monde sont alors actives sur le marché des calculatrices électroniques: aux fabricants spécialisés comme Busicom, Systec et Casio s'ajoutent des manufacturiers de produits électroniques comme Hitachi, Toshiba, Sony et Ricoh, qui se lancent eux aussi dans la bataille de la calculatrice électronique.

En 1965, Casio présente sa première calculatrice de table entièrement électronique, le modèle 001, et cesse progressivement la fabrication de calculatrices à relais. Le marché des calculatrices électroniques est dominé par la guerre des prix, lesquels ne cesseront dès lors de chuter. En cinq ans, le prix initial de 2500 $, correspondant à celui du premier modèle électronique de Sharp, est divisé par 10. OMRON est la première compagnie à offrir dès 1971 une calculatrice à un prix inférieur à 250 $.

Le développement du circuit intégré MOS bouleverse l'évolution des calculatrices. Des circuits de plus en plus compacts permettent de réduire drastiquement la taille des calculatrices et leur consommation électrique. Les machines deviennent alors portables. Avec l'AS-8, Casio offre un produit qui, avec un prix inférieur à 200 $, défie la concurrence et catapulte l'entreprise en tête des fabricants. Le marché est florissant, mais les prix sont encore trop élevés pour les besoins individuels. La course pour la mise au point d'une calculatrice à un prix inférieur à 50 $ débute, et Casio se doit de la mener afin de survivre. La Casio Mini de poche introduite en 1972 remporte un vif succès avec10 millions d'unités vendues en une année (fig. 3.3).

Avec des circuits toujours plus denses, une calculatrice ne demande bientôt plus qu'un seul circuit. La fabrication se limite ainsi au simple assemblage de ce circuit avec un clavier et un affichage. Chacun peut alors se lancer dans la production de calculatrices électroniques portables.

Fig. 3.3 Une Casio Mini (1972) (photo Guy Ball).

De nombreuses entreprises abandonnent le secteur des calculatrices électroniques durant l'année 1974. La demande est toutefois en hausse constante et les fabricants y répondent en offrant de meilleurs prix et de meilleures performances. Casio continue à participer à ce développement, mais se lance parallèlement sur un autre marché, celui de la montre électronique.

Premières calculatrices électroniques

L'innovation se loge généralement dans une forme originellement conçue pour un autre fonctionnement. A l'instar des premières automobiles ressemblant à des diligences, les premières calculatrices électroniques se présentent avec tous les aspects des machines électromécaniques.

Au début des années 1960, une activité importante règne au Japon et aux Etats-Unis afin de faire évoluer les calculatrices électromécaniques et de les remplacer par des machines transistorisées. Les caractéristiques visées sont un fonctionnement silencieux, une manipulation simple, une fonctionnalité accrue et un produit meilleur marché. De nombreux fabricants, parmi lesquels Canon et Sharp, se fixent ces objectifs. Le résultat sera une machine constituée de transistors discrets, pesant près de 20 kg et coûtant plusieurs milliers de dollars.

Anita (*A New Inspiration To Arithmetic*) est considérée comme la première calculatrice électronique. Elle est fabriquée dès 1961 en Angleterre par Sumlock Comptometer, un important fabricant de calculatrices électromécaniques (fig. 3.4).

En 1964 apparaît une machine construite par Friden et utilisant un tube cathodique comme affichage. Sony lance un premier modèle électronique baptisé MD-5, et dont l'affichage est composé de tubes à décharge gazeuse appelés tubes Nixie. Sharp annonce au Japon la naissance de sa première calculatrice baptisée Compet (CS-10A).

Fig. 3.4 Anita, la première machine à calculer électronique (1962) (photo Frank Boehm).

Ces premières machines sont des calculatrices à quatre opérations et nombres flottants et dont le prix oscille entre 1000 et 2500 $. Bien que plus chères que les machines mécaniques, les machines électroniques présentent de nombreux avantages: calcul rapide, emploi simple, fonctionnement silencieux et maintenance très réduite.

Wang

La puissance de calcul offerte par l'électronique est rapidement mise au service de machines capables d'effectuer des opérations complexes, notamment des transformations trigonométriques ou logarithmiques. Wang remporte un certain succès avec LOCI, une des premières machines à calculer électronique à usage scientifique, commercialisée dès 1964 par son entreprise, Wang Laboratories. Celle-ci sera néanmoins détrônée par la calculatrice Hewlett-Packard 9100.

L'inventeur de LOCI, le Dr An Wang, n'est pas un néophyte en matière d'ordinateurs. Emigrant chinois arrivé aux Etats-Unis en 1945, il entre à l'Université de Harvard où il travaille avec Aiken, le grand constructeur des ordinateurs de la famille Mark. Il se distingue peu après sa thèse de doctorat par l'invention des tores de ferrite en 1949, et dépose un brevet intitulé *Pulse Transfer Controlling Device*. L'invention concerne la possibilité de magnétiser un tore de ferrite selon une direction ou l'autre, en le mettant dans un état égal à 0 ou 1. La principale propriété associée à ce matériau est la possibilité de le modifier électriquement et d'en faire une cellule de mémoire de 1 bit. Cette invention a une signification considérable. Le tore de ferrite constituera pendant longtemps les mémoires des ordinateurs et trouvera aussi une place privilégiée dans les vaisseaux spatiaux en raison de son insensibilité aux rayons cosmiques.

Pour l'anecdote, Wang profite de la politique de l'Université de Harvard en matière de brevets, à savoir son désintéressement total envers toute invention à usage commercial. Il devient donc propriétaire de sa découverte, avant d'en revendre les droits à bon prix quelques années plus tard à IBM.

LOCI est un calculateur scientifique de table (fig. 3.5). Son nom provient de *Logarithmic Calculating Instrument*, indiquant que les calculs effectués se basent sur l'utilisation des logarithmes. Les circuits multiplicateurs étant très chers à l'époque, Wang développe une méthode pour effectuer les multiplications et les divisions par l'intermédiaire des opérations plus simples que sont l'addition et la soustraction. Les logarithmes sont en effet des transformations particulières qui convertissent notamment une multiplication en une somme. Ainsi, le produit $a \times b$ est calculé par la somme $\log(a)+\log(b)$. Ceci requiert toutefois de recourir à des fonctions de transformation log et antilog, réalisées sur LOCI par du matériel spécialisé. Les calculateurs Wang sont donc capables de générer très rapidement les fonctions logarithmiques et exponentielles. Les autres fonctions, comme celles liées à la trigonométrie et réalisées de manière conventionnelle par microprogramme, sont quant à elles beaucoup plus lentes à exécuter.

Fig. 3.5 LOCI (vers 1964) (photo Bob Roswell).

LOCI, vendu au prix de 6700 $, est plus un mini-ordinateur qu'une calculatrice. Son successeur, Wang 300, lancé en 1965, est quant à lui une véritable calculatrice comprenant environ 300 transistors. Il est composé d'une unité de table avec un clavier; son prix fixé initialement à 1700 $ tombe à 600 $ six ans plus tard.

Après avoir dominé le marché des calculatrices électroniques de table durant la deuxième moitié des années 1960, Wang Laboratories se trouve bientôt confronté à la rude concurrence des produits à base de puces. L'entreprise décide alors de réorienter ses efforts vers le domaine du traitement de textes.

HP

Produite par Hewlett-Packard, la HP9100 arrive en 1968 [9]. C'est une calculatrice de table pour scientifiques et ingénieurs qui offre une vaste palette de fonctionnalités inédites.

La calculatrice HP9100 propose une notation scientifique, un domaine de représentation allant de 10^{-99} à 10^{99}, des fonctions logarithmiques, trigonométriques et hyperboliques ainsi que leurs inverses; s'y ajoutent l'addition et la soustraction vectorielles, ainsi que la conversion polaire/rectangulaire. Elle comporte un affichage vert à trois lignes implémenté par un tube cathodique, un lecteur de cartes magnétiques et une imprimante ou un traceur en option (fig. 3.6).

La machine avoisine les 20 kg, pour une consommation de 70 W. Vendue à 4900 $, et bien que plus chère que certaines automobiles de l'époque, elle demeure nettement meilleur marché que les mini-ordinateurs et rencontre un vif succès.

Le nom de l'entreprise HP provient de ses deux fondateurs, Dave Packard et Bill Hewlett. La légende veut qu'il ait été joué à pile ou face en janvier 1939 par les deux jeunes ingénieurs fraîchement diplômés de l'université de Stanford, et que Bill Hewlett ayant gagné, il pu mettre son nom en premier. Cette raison sociale fut définitivement établie à l'occasion de l'achat par les studios Walt Disney de 9 oscillateurs audio de modèle 200A, la première commande importante passée à cette jeune entreprise. La firme se développe d'abord dans l'instrumentation électronique. Dans les

Fig. 3.6 Le modèle HP9100 (1968) (The Museum of HP Calculators).

années 1950, ce sont près de vingt nouveaux instruments qui sont introduits chaque année. L'entreprise progresse grâce à la direction forte et clairvoyante de ses deux fondateurs, mais aussi grâce à une politique de participation d'avant-garde qui implique les employés dans les succès ou les ralentissements économiques de la société.

Quand ses ingénieurs présentent la calculatrice HP9100 à Bill Hewlett, celui-ci leur suggère de concevoir une machine pouvant se glisser dans la poche d'une chemise. Les objectifs du successeur de la HP9100 que l'équipe doit atteindre sont une vitesse dix fois supérieure, un volume dix fois inférieur et un coût dix fois moindre. Ces caractéristiques de prix et de volume ne seront que partiellement concrétisées avec la HP-35, mais presque réalisées avec la HP-65. La vitesse désirée ne sera cependant approchée qu'avec les machines de table alimentées sur secteur.

Kilby & Noyce

L'électronique passe progressivement de l'emploi des transistors à celui des circuits intégrés.

C'est sans se connaître au préalable et en suivant des chemins indépendants que Kilby et Noyce inventent presque simultanément le circuit intégré.

Jack St Clair Kilby naît dans le Missouri en 1923. Après des études d'ingénieur électricien et dix ans d'activité de recherche, il entre chez Texas Instruments à Dallas au Texas en 1958. Robert Norton Noyce, lui, voit le jour en 1927 dans l'Iowa. Docteur physicien diplômé du MIT, il travaille quelques années dans une entreprise californienne, avant de participer avec d'autres partenaires à la fondation de Fairchild Semiconductor Corporation en 1957.

Le transistor existe alors depuis 10 ans et est utilisé dans de nombreux appareils, notamment les radios. Le marché est florissant. Mais d'autres perspectives apparaissent avec l'électronique numérique. Les ingénieurs sont conscients de cette possibilité, mais savent que les produits à venir exigeront un nombre toujours plus grand de composants et qu'il sera difficile de faire face à ce besoin accru. Les techniques de fabrication de l'é-

poque ne permettent par ailleurs pas d'augmenter le nombre de composants assemblés de manière fiable. C'est pourquoi plutôt que de fabriquer une grande quantité de transistors discrets et de les assembler par soudage, les ingénieurs tentent de réaliser directement un réseau de transistors interconnectés. Kilby et Noyce travaillent sur ce projet.

La solution sera le circuit intégré monolithique (fig. 3.7). Les deux ingénieurs découvrent simultanément un moyen de fabriquer en une seule fois l'ensemble des transistors interconnectés sur un seul cristal de matériau semiconducteur: la puce. Kilby utilise le germanium en guise de semiconducteur, alors que Noyce choisit le silicium.

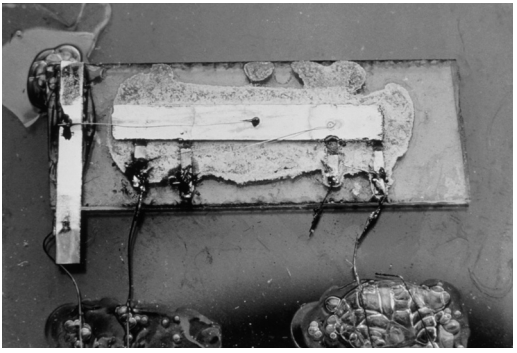


Fig. 3.7 Le premier circuit intégré chez Texas (1958) (Texas Instruments Incorporate).

Des brevets sont déposés en 1959 à la fois par Texas Instruments et Fairchild Semiconductor. La bataille juridique opposant les deux firmes durera près d'une dizaine d'années avant que les protagonistes ne décident de partager les licences de leurs technologies. L'invention de la puce par Kilby et Noyce est reconnue comme l'une des innovations majeures de l'histoire de l'humanité. Elle jouera un rôle déterminant dans l'évolution des calculatrices électroniques.

Les calculatrices japonaises

Inventé et maîtrisé aux Etats-Unis, le circuit intégré donne tous les atouts à cette nation pour s'imposer sur le marché des machines à calculer mais c'est sans compter la puissance commerciale considérable du Japon.

Si les Etats-Unis disposent d'une avance manifeste dans la technologie des circuits électroniques, le Japon peut se prévaloir de sa suprématie dans le domaine commercial des calculatrices. Les progrès de l'électronique bouleverseront ce marché en quelques années seulement.

Sur le plan technique, on assiste au fil des ans à une complication des circuits intégrés (CI) et donc à une diminution progressive du nombre de circuits nécessaires à la réalisation d'une machine à calculer. Cette réduction est singulièrement illustrée par

les machines électroniques successives de Sharp. La machine de 1964 comprend des transistors discrets, celle de 1966 145 CI, 5 seulement en 1969 et un seul en 1972.

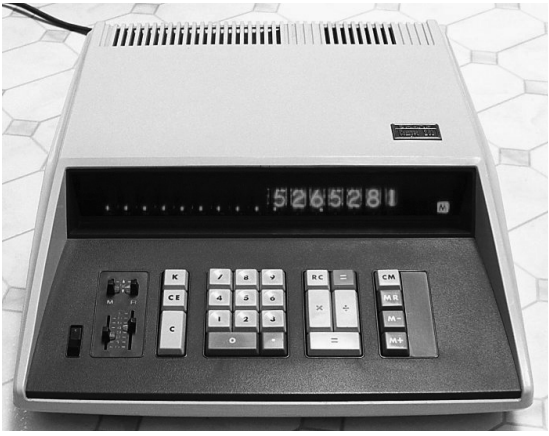


Fig. 3.8 Un modèle Sharp à 4 opérations (fin des années 1960) (The Museum of HP Calculators).

Sur le plan commercial, les machines électroniques performantes remplacent progressivement les calculatrices électromécaniques. Les prix font la loi. Très élevé au départ, celui des machines électroniques tombe en chute libre pour atteindre en 1969 des niveaux inférieurs à ceux des machines électromécaniques. L'événement historique qui symbolise ce passage est la présentation à New York des nouveaux produits de l'entreprise japonaise Sharp. Dans sa palette de machines traditionnelles et électroniques, Sharp offre pour la première fois une calculatrice électronique de table à quatre opérations nommée QT-8, meilleur marché que son équivalent traditionnel vendu à 1000 $ (fig. 3.8). Ce prix très compétitif s'explique par une électronique composée de 5 circuits intégrés seulement.

Notons que les CI de cette calculatrice étaient fabriqués sous contrat par North American Rockwell Microelectronics. Cette alliance entre un fabricant de calculatrices japonais et un fabricant de composants américain est caractéristique de l'époque. Les firmes japonaises Canon et Ricoh ont alors American Microsystems comme partenaires, Seiko optant elle pour Signetics.

TI et les calculatrices de poche

La réduction progressive du nombre de circuits intégrés mène à une diminution concomitante de la taille des calculatrices. Cette miniaturisation, conduite en parallèle avec la maîtrise de l'alimentation électrique autonome, débouchera sur l'apparition des calculatrices de poche et leur explosion sur le marché.

Jack Kilby, l'inventeur du CI, travaille chez Texas Instruments (TI). Associé à Jerry Merryman et James Van Tassel, il invente formellement la calculatrice de poche en déposant un brevet en septembre 1967. L'élément central de cette innovation consiste en l'utilisation d'un seul circuit intégré pour effectuer toutes les opérations logiques nécessaires au calcul. Les inventeurs parlent d'un «circuit à haute densité d'intégration» (LSI) bien que celui-ci ne comporte qu'un millier de transistors. Les trois inventeurs ne réalisent alors pas l'importance que les calculatrices mono puce allaient avoir jusqu'à aujourd'hui. «Notre projet visait surtout à démontrer le potentiel des CI et à leur ouvrir un marché» rapporte Jack Kilby [13].

Certaines difficultés demeurent toutefois, notamment liées à la qualité des wafers de deux pouces de l'époque, à l'encapsulation des CI, à la puissance limitée des piles et au choix de l'affichage. Ce n'est que trois ans plus tard que le LSI rendra possible une machine à quatre opérations, comportant des nombres à 12 chiffres avec placement automatique de la virgule et un affichage par imprimante thermique. En 1970, le fabricant japonais Canon propose sur le marché japonais un Pocketronic à 400 $ contenant les circuits de Texas Instruments.

La livraison de calculatrices de poche avec la marque TI ne commence qu'en 1972 avec la TI-2500. Cette machine propose les 4 opérations et un affichage à diodes électroluminescentes, pour un prix de 120 $.

La mort de la règle à calcul

Les calculatrices électroniques de poche ont tant d'avantages qu'elles s'imposent rapidement dans les bureaux, les écoles et à la maison. Leur arrivée marque la disparition des outils de calcul portables de l'époque que sont la règle à calcul et la petite machine à calculer mécanique Curta (fig. 3.9c).

En 1972 arrive la HP-35, première calculatrice de poche à offrir les fonctions logarithmiques et trigonométriques. Elle jouit d'un grand succès auprès des écoliers et des professions techniques en tant que calculatrice scientifique de poche. Elle offre pour la première fois et dans un petit format le calcul et l'affichage des nombres en notation scientifique avec une dynamique opérationnelle de 200 décades, ainsi que l'accès aux fonctions scientifiques usuelles que sont les fonctions inverse, racine, logarithmique et trigonométrique. Le temps de calcul est de l'ordre 0,5 seconde pour une précision au 10^e chiffre significatif. Les modèles suivants sont la HP-45, plus puissante, et la HP-65, programmable. Avec cette dernière, la séquence des touches pressées peut être mémorisée et rejouée par la suite. Le programme peut comporter jusqu'à 100 pas. Une fois enregistré, il peut aussi être copié sur une carte magnétique externe pour le stockage et une utilisation ultérieure.

Face à cette innovation, la règle à calcul périclite. TI reprend d'ailleurs ce nom (Slide Rule, ou SR-10) pour baptiser sa première calculatrice de poche vendue en grande quantité, afin de souligner cette transition. Quant aux producteurs de règles à

calcul, ils se reconvertissent: Aristo, Denner et Pape proposent en 1972 une calculatrice à quatre opérations construite autour d'un circuit de TI. La fabrication des règles à calcul cessera définitivement en 1975.

Fig. 3.9 Changement des moyens de calcul: (a) Règle à calcul (b) Curta (c) HP-35. (The Museum of HP Calculators)

La notation polonaise inverse

Dès le départ, les utilisateurs de machines à calculer électroniques sont partagés sur le choix du type de machine. Entre adeptes des machines pourvues de la touche ENTER et partisans des machines qui en sont dépourvues, la guerre va bon train.

Pour additionner huit et six sur une calculatrice ordinaire utilisant la notation algébrique directe, vous pressez successivement:

[8] [+] [6] [=]

Sur une calculatrice pourvue de la touche ENTER et appliquant donc la notation polonaise inverse, le principe consiste à entrer les opérants avant l'opérateur.

Vous pressez donc:

[8] [ENTER] [6] [+]

pour que les 8 et 6 soient disponibles en mémoire au moment de presser le +. La touche ENTER joue ce rôle particulier et est caractéristique des calculatrices utilisant cette notation.

A l'instar de Wang, Hewlett-Packard utilise la notation polonaise inverse (NPI) sur toute sa gamme de calculatrices. Deux raisons à ce choix: d'une part, la NPI est une manière efficace pour la logique interne d'exécution des opérations. D'autre part, elle permet à un utilisateur technicien (public cible des calculatrices scientifiques) d'effectuer des calculs complexes en pressant peu de touches.

☞ Voici à titre d'exemple, les séquences à presser pour introduire l'expression {(8+6)(7-5)}/(9-7) dans la machine. En notation traditionnelle et sur une machine pourvue d'une mémoire, la séquence est:

[8] [+] [6] [=] [M+] [7] [-] [5] [x] [MR] [=]

[MC] [M+] [9] [-] [7] [/] [MR] [=]

[MC] [M+] [1] [/] [MR] [=]

Toujours en notation traditionnelle mais avec une machine pourvue de parenthèses, la séquence est:

[(] [(] [8] [+] [6] [)] [*] [(] [7] [-] [5] [)] [)] [/] [(] [9] [-] [7] [)] [=]

Selon la NPI, on tape la séquence:

[8] [ENTER] [6] [+] [7] [ENTER] [5] [-] [x] [9] [ENTER] [7] [-] [/]

Cela fait 25 touches pressées dans le premier cas, 20 dans le deuxième, contre 14 dans le dernier cas. On constate alors l'économie que procure la NPI à cet égard.

On doit la notation polonaise inverse au mathématicien polonais Jan Lukasiewicz qui démontra que les parenthèses ne sont pas nécessaires si l'opération est spécifiée avant les opérants (notation polonaise ou préfixe) ou après les opérants (notation polonaise inverse ou postfixe). La nature séquentielle des opérations à effectuer par la calculatrice, c'est-à-dire la mise en pile des opérants suivie de l'exécution de l'opération parle naturellement pour le choix de la NPI.

Calculatrices et microprocesseurs

L'apparition des circuits intégrés a conduit à l'explosion des ventes de machines à calculer: une invention a stimulé le marché. Mais l'inverse se produit lorsque le microprocesseur voit le jour.

En 1969, le fabricant japonais de calculatrices Busicom demande à Intel de concevoir un jeu de CI destinés à une famille de calculatrices de table programmables. Intel est choisi parce que, concurrence oblige, il n'est le partenaire d'aucun autre fabricant japonais, notamment en raison de la jeunesse de l'entreprise fondée une année auparavant par Robert N. Noyce (qui vient de quitter Fairchild) et Gordon E. Moore. L'objectif majeur à long terme d'Intel est le développement des circuits semi-conducteur de mémoire. Le cahier des charges de Busicom prévoit une douzaine de CI dont certains nécessitent l'intégration de plus de 5000 transistors sur une puce. C'est un défi pour Intel, qui réalise alors des composants mémoires d'une capacité d'environ 2000 bits.

Le projet est confié à Ted Hoff, connu pour sa conviction qu'en tout, «il doit y avoir une meilleure solution» [4]. Familier des ordinateurs après des travaux menés sur le mini-ordinateur PDP-8 de Digital, il esquisse une solution inspirée de ces recherches. Il prévoit un circuit chargé des calculs, un autre pour mémoriser le programme, un troisième pour mémoriser les données et finalement un quatrième destiné à communiquer avec la périphérie. Cette solution présente plusieurs avantages. En plus de remplacer les 12 puces initialement prévues par Busicom par quatre seulement, ce schéma d'organisation limite le nombre de transistors par puce de 5000 à 2500. Cette proposition brillante remporte l'agrément de Busicom qui mandate Intel pour le développement. Un an plus tard, les circuits sont produits. Sans en porter encore le nom, le premier microprocesseur vient de naître.

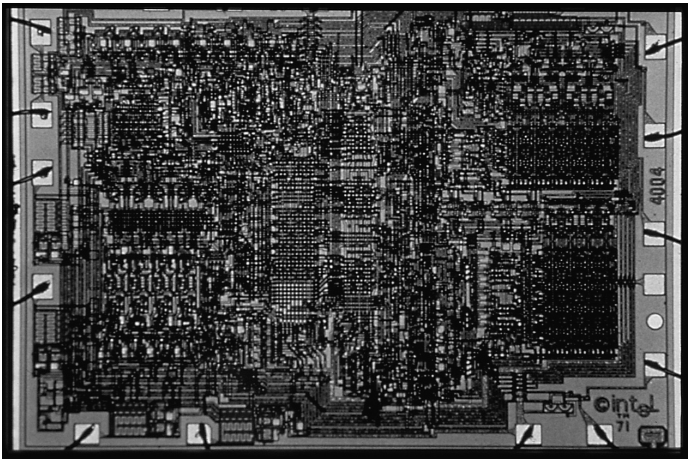


Fig. 3.10 Intel 4004 (Intel).

Le contrat entre Busicom et Intel réservait au mandataire Busicom les droits exclusifs sur le jeu de CI. Peu conscient du potentiel et du réel enjeu de ce qui venait d'être développé, celui-ci cède à Intel ses droits en échange d'une réduction des coûts de fabrication. En 1971, Intel nomme son microprocesseur Intel 4004 et le lance sur le marché, avec le succès que l'on sait (fig. 3.10).

Calculer avec des transistors

S'il est relativement simple de comprendre comment compter avec des rouages mécaniques à dix positions, il est plus difficile d'appréhender la manière dont on compte grâce à des transistors.

Le passage des machines à calculer mécaniques aux machines électroniques ne va pas sans un profond remaniement du principe de fonctionnement des calculatrices. La règle de base des machines mécaniques consistait à faire tourner un ensemble de roues; celles-ci sont abandonnées et remplacées par un assemblage de commutateurs interconnectés. Les opérations sont désormais effectuées par logique binaire. Le calcul à base de transistors est donc une formalisation des opérations par les opérateurs binaires inventés par Georges Boole.

Un commutateur est un simple contact entre deux bornes, commandé par un signal appliqué sur une troisième borne. Le contact est ouvert ou fermé, suivant l'état du signal. La forme électromécanique conventionnelle du commutateur, telle que les interrupteurs que nous actionnons pour allumer une lampe, n'est pas idéale pour réaliser des machines à calculer car elle combine le contact électrique avec une commande mécanique. L'idéal est un commutateur à commande électrique, où l'état ouvert ou fermé dépend d'un signal électrique. Historiquement, la commutation électrique apparaît avec le commutateur téléphonique inventé par Almon Strowger en 1891. Commandé électriquement, ce système demeure néanmoins électromécanique, car des déplacements mécaniques subsistent. La suppression de tout mouvement mécanique naît avec la commutation électronique. Son histoire débute avec les tubes à vide, dont on attribue la paternité à Lee De Forest. Il invente en 1906 l'audion, premier tube à vide possédant un signal de commande qui agit sur le courant de sortie à travers une grille disposée à l'intérieur du tube. L'évolution de la commutation s'accélère ensuite avec l'apparition du transistor en 1947.

Les transistors fonctionnent comme des interrupteurs commandés électriquement. Pour additionner avec ces commutateurs, on applique la règle des retenues. La fonction binaire élémentaire «Add» qui en résulte comporte 3 entrées et 2 sorties et peut être réalisée grâce à un circuit à 18 commutateurs (fig. 3.13).

☞ Abordons la description détaillée de l'addition de deux nombres binaires, en commençant par une analyse du fonctionnement d'un commutateur.
Les commutateurs de la figure 3.11 font apparaître un contact horizontal disposé entre les bornes gauche x et droite y ainsi que sa commande par le signal a. On distingue deux sortes de commutateurs, différant dans leur fonctionnement. L'un (a) est fermé au repos. Son contact est fermé quand a = 0 et ouvert quand a = 1. Le commutateur ouvert au repos (b) fonctionne de manière complémentaire. Notons encore que la sortie y du commutateur prend la valeur y = x de l'entrée quand le contact est fermé et qu'il est indéterminé dans le cas contraire.
L'addition de nombres décimaux se fait de manière très ordinaire. Elle considère les nombres disposés selon la figure 3.12, et a = 89 et b = 106 ont été choisis en guise d'exemples. Chaque colonne est traitée successivement en procédant de droite à gauche. Dans chacune d'elles, les chiffres sont additionnés et la somme est reportée dans la somme s, avec inscription éventuelle d'un report r dans la colonne de gauche en cas de somme à deux chiffres.
L'addition des nombres binaires procède de manière similaire (fig. 3.13). On procède là aussi de droite à gauche, avec une addition par colonne. Un report est produit en cas de dépassement.

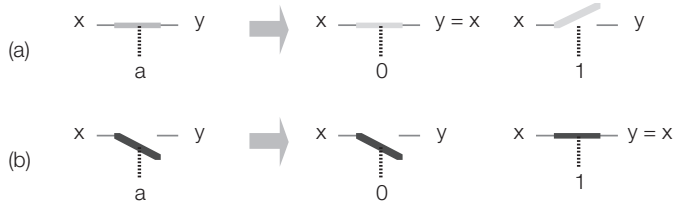


Fig. 3.11 Le transistor vu comme un commutateur: (a) commutateur fermé au repos; (b) commutateur ouvert au repos

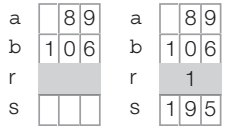


Fig. 3.12 Addition de nombres décimaux

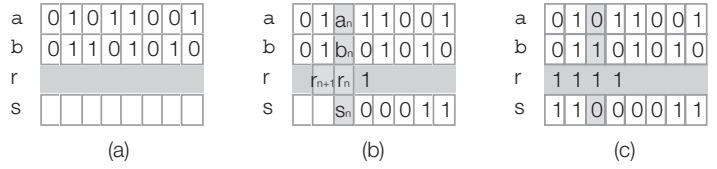


Fig. 3.13 Addition de nombres binaires (a = 89, b = 106).

Cette addition par colonne peut être formalisée par une simple fonction faisant intervenir ces cinq éléments, soit a_n, b_n et r_n en entrée, et les éléments somme s_n et report r_{n+1} associés en sortie. L'indice n signifie que la fonction est valable pour chacune des colonnes n. Cette fonction est appelée additionneur complet à 1 bit et elle est représentée par le module Add dans la figure 3.14 (a). Grâce à son caractère binaire, elle peut être aisément réalisée par des commutateurs assemblés selon la réalisation illustrée en 3.14 (b). Un schéma de l'activation des commutateurs pour la situation de la colonne grisée de la figure 3.13 est donné en 3.14 (c).

L'addition complète de deux nombres binaires requérant un usage répété de l'addition par colonne, sa réalisation nécessitera autant de blocs Add que de colonnes à additionner.

☞ Il faut par conséquent compter autant de blocs que de bits dans les nombres à additionner. Par exemple, la réalisation d'un additionneur ADD à 4 bits peut se concrétiser avec 4 blocs Add, qu'il convient d'interconnecter de façon à transmettre les reports de colonne en colonne (fig. 3.15).
Une fois réalisé avec des transistors selon le schéma proposé, ce circuit ADD reçoit en entrée les deux nombres à 4 bits **a** et **b** et produit directement en sortie leur somme **s** = **a** + **b**.

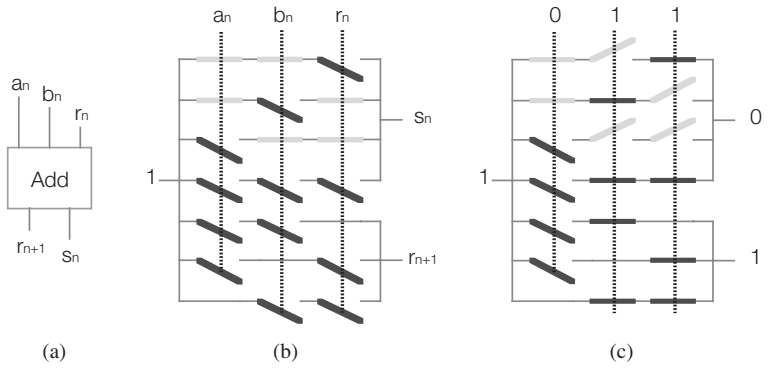


Fig. 3.14 Réalisation de l'additionneur élémentaire par des commutateurs.

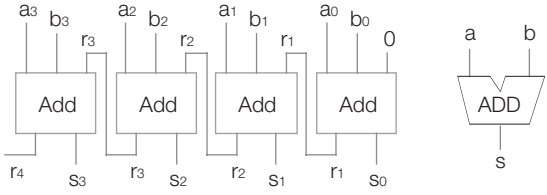


Fig. 3.15 Additionneur à 4 bits.

Une calculatrice de poche

L'additionneur est l'élément de base d'une machine à calculer, mais d'autres unités électroniques, un clavier et un affichage sont nécessaires afin d'obtenir une machine à calculer électronique complète.

Les autres unités requises au sein d'une machine à calculer concernent notamment les tâches de séquencement des opérations, d'exécution d'autres fonctions mathématiques et de dialogue avec l'utilisateur à travers le clavier et l'affichage.

La figure 3.16 illustre la structure du CI qui constitue le cœur d'une simple calculatrice de poche [5]. Outre l'additionneur noté ADD, ce CI comprend un ensemble de registres destinés à mémoriser les nombres, ainsi qu'un séquenceur assurant le déroulement des opérations dictées par la suite d'instructions stockées en mémoire.

Les 7 registres dénommés X, Y, Z, T, W1, W2 et M sont des mémoires pour les nombres. Comme chaque registre est constitué de 14 cases et que chacune d'entre elles peut recevoir un chiffre décimal, la machine a une capacité de représentation des nombres à 14 chiffres. Chaque registre peut donc recevoir un nombre décimal com-

posé d'un maximum de 14 chiffres. Les registres remplissent des fonctions bien définies. X, Y, Z et T reçoivent les opérants, M est le registre de mémorisation, W1 et W2 sont des registres de travail, utiles pour mémoriser des résultats intermédiaires durant le déroulement du calcul.

L'additionneur ADD est construit et fonctionne selon le principe général énoncé plus haut (fig. 3.14). L'unité illustrée est dotée de la double fonctionnalité de l'addition et de la soustraction. Les liaisons établies entre cette unité et les 7 registres montrent que l'additionneur opère seulement sur deux chiffres à la fois, qui proviennent de deux registres distincts. Elle effectue donc une addition ou une soustraction sur les deux chiffres encodés en binaire qui lui arrivent par la droite et produit un résultat vers la gauche. L'unité ne remplit ainsi qu'une partie de l'addition de deux nombres, c'est-à-dire l'addition par colonne élémentaire évoquée plus haut. Conformément à la méthode énoncée, l'addition complète requiert encore la répétition de l'addition par colonne élémentaire pour chacun des 14 chiffres des nombres à additionner. Une partie répétitive vient donc s'ajouter. Elle est réalisée par une suite de 14 pas successifs dirigés par les instructions du séquenceur. La fonction d'addition de deux nombres est réalisée par une utilisation répétée de l'additionneur placée sous le contrôle des instructions de séquencement.

Une partie des instructions du séquenceur est ainsi consacrée au déroulement de l'opération d'addition. D'autres instructions remplissent diverses autres fonctions. L'une d'elles est par exemple la mise en mémoire M par copie du registre X dans le registre M. Une fonction plus complexe est notamment celle assurant la multiplication de X par Y. Chaque fonction fait ainsi l'objet d'une séquence bien précise, parmi les 768 instructions que comporte le microprogramme de la machine.

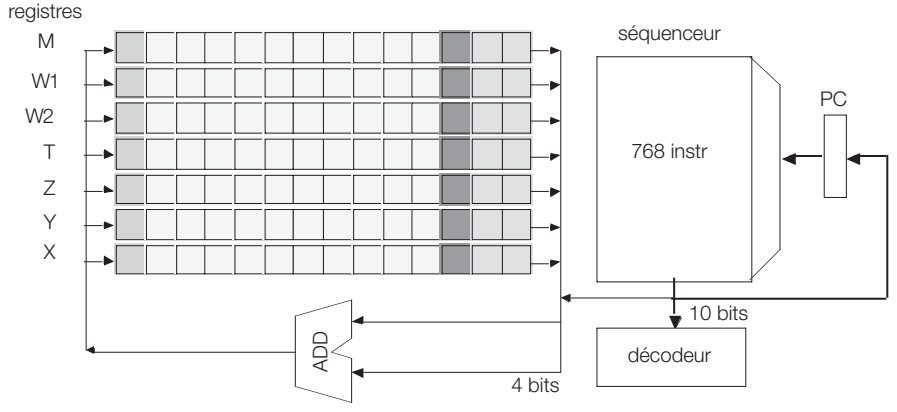


Fig. 3.16 Architecture du CI au cœur d'une calculatrice de poche.

Outre le CI placé au cœur de la machine, la calculatrice comprend un élément actif, la pile, ainsi que des éléments externes: un clavier et un affichage. La figure 3.17 offre un schéma d'interconnexion pour une calculatrice conventionnelle équipée d'un clavier et d'un affichage à 7 segments. Le CI est relié avec les touches du clavier et avec chaque chiffre de l'affichage selon un schéma matriciel qui limite le nombre de fils de connexion. L'activation de chaque chiffre de l'affichage et la lecture de chaque touche du clavier interviennent ainsi à intervalles réguliers selon l'activation sélective de lignes de balayage.

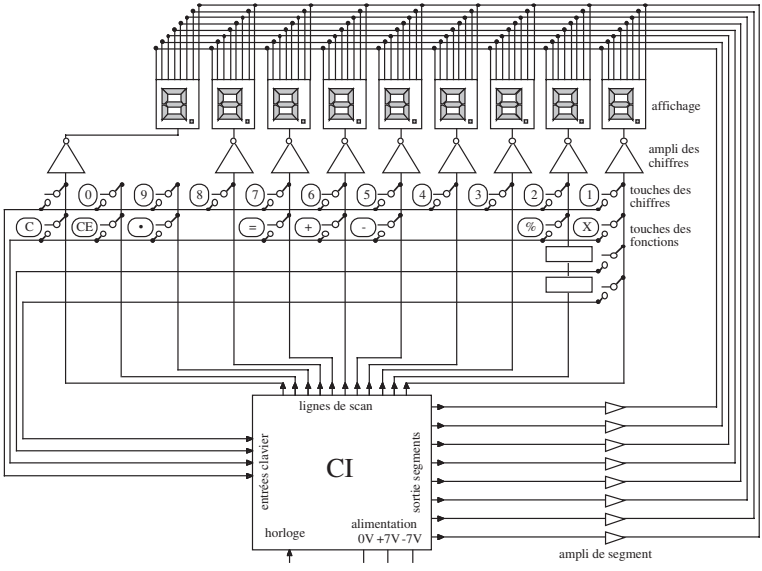


Fig. 3.17 Circuit de calculatrice avec une seule puce.

Les concepts

L'histoire des machines à calculer est dominée par le passage des machines mécaniques et électromécaniques vers leur substitut électronique. Les calculatrices électromécaniques ont atteint un bon niveau de développement dans les années 1950 et n'ont pas été immédiatement dépassées par les premières machines électroniques. L'invention du transistor en 1947 lance la course aux ordinateurs, mais cet événement n'aura pas d'impact immédiat sur l'évolution des calculatrices électroniques car les modèles qu'il permet de réaliser ne sont pas compétitifs. C'est l'invention du circuit intégré en 1958 qui constituera à ce titre l'étape déterminante. Il représente la technologie de base qui permettra de réaliser des calculatrices compactes et fiables, et qui ouvrira la voie aux calculatrices de table commerciales et portables.

Comparées à leurs ancêtres électromécaniques, les calculatrices électroniques présentent de nombreux avantages, notamment en matière de fiabilité et de rapidité de calcul, mais aussi quant à leur taille, leur confort d'utilisation et bien entendu leur coût. Mais c'est plus particulièrement pour les fonctionnalités offertes pour le calcul scientifique que les machines électroniques se démarquent. La calculatrice de table HP9100 et son successeur portable HP-35 sont à ce titre demeurés deux produits exemplaires de l'évolution des machines à calculer.

Le succès commercial des calculatrices portables sonne le glas des règles à calcul et stimulera la recherche, dont l'une des réalisations décisives sera l'invention du microprocesseur en 1971.

Les collectionneurs [2] [4] [7] et les musées [3] [6] [9] [11] [12] montrent actuellement un grand intérêt pour les premières machines à calculer électroniques.

Références

[1] André Rochat, «Histoire de la bureautique et de l'informatique», *Flash informatique*, Juillet 1998. http://sawww.epfl.ch/SIC/SA/publications/FI98/fi-7-98/7-98-page1.html

[2] Computer History Association of California. http://www.chac.org/chhistpg.html

[3] Computer Museum of America. http://www.computer-museum.org/

[4] James Redin, «Vintage Calculators». http://www.dotpoint.com/xnumber/vintage.htm

[5] E. W. McWhorter, «The Small Electronic Calculator», *Scientific American*, March 1976, Vol. 234, N° 3, pp 88-98.

[6] Obsolete Computer Museum. http://www.obsoletecomputermuseum.org

[7] Rick Furr, The Calculator Reference. http://www.vcalc.net/

[8] Tadao Kashio, «Creativity and Contribution». http://world.casio.com/corporate/company/history.html

[9] The Museum of HP Calculators. http://www.hpmuseum.org/

[10] The Virtual Museum of Computing. http://vmoc.museophile.com/

[11] University of Virginia, Computer Science Department, Computer Museum. http://www.cs.virginia.edu/brochure/museum.html

[12] Vintage calculators http://www.vintagecalculators.com/

[13] Irene Kim, «Function at Your Fingerprints.» Mechanical Engineering Magazine. Vol. 112, N° 1. January 1990.

CHAPITRE 4

HISTOIRE DES ORDINATEURS

«Je pensais que les ordinateurs seraient une idée applicable universellement, comme les livres. Mais je n'ai jamais pensé qu'ils se développeraient si vite, parce que je n'ai jamais deviné que les circuits intégrés comporteraient si vite autant d'éléments. Le transistor a été découvert sans que l'on s'y attende. Tout ceci est arrivé bien plus vite que prévu.»
J. Presper Eckert, co-inventeur de l'ENIAC, en 1991

L'invention des ordinateurs

Le terme français «ordinateur» a été proposé pour la première fois à IBM France par Jacques Perret, professeur à la Sorbonne, tandis que tous les autres pays choisissaient le terme anglais de «computer» (calculateur). Un ordinateur est toutefois très différent d'une machine à calculer: il comporte un programme, suite d'ordres ou d'instructions lui indiquant ce qu'il doit faire, enregistrés dans l'ordinateur. Une fois son programme mémorisé, l'ordinateur est autonome et peut exécuter les instructions sans que quelqu'un ne tape préalablement sur un clavier les données qu'il doit utiliser. Il peut par ailleurs se brancher sans difficulté à une autre partie du programme. Une autre caractéristique de l'ordinateur qui semble à tort être la plus importante, est d'être électronique; ceci n'agit toutefois que sur la vitesse de la machine, et non sur sa fonctionnalité.

L'ordinateur est généralement présenté comme une unité de calcul capable de réaliser des opérations telles que des additions et des multiplications en base 2 comme une simple calculatrice électronique… c'est effectivement le cas, mais celle-ci réalise aussi ces opérations sans être pour autant un ordinateur.

Un ordinateur comporte une unité de commande capable de traiter un programme, ainsi qu'une unité de traitement pouvant réaliser des opérations sur des nombres. Le programme de l'application peut soit être enregistré dans l'unité de commande soit être fourni de l'extérieur, par exemple au moyen de cartes perforées (fig. 4.1).

L'unité de traitement réalise des opérations analogues à celle d'un boulier (fig. 4.2): un premier nombre est sélectionné dans la mémoire de données et chargé dans le registre 1, alors qu'un deuxième nombre est chargé dans le registre 2. Ceci équivaut à choisir les deux nombres à additionner sur le boulier en plaçant correctement les boules correspondantes. Les données sont alors envoyées à l'unité de calcul appelée ALU (Arithmetic and Logic Unit) pour être additionnées, et le résultat est placé dans le deuxième registre. C'est aussi ce que doit faire l'utilisateur d'un boulier qui rassemble les boules représentant ses deux nombres, avant de pouvoir lire le résultat final. Une même unité de traitement équipe aussi une simple calculatrice de poche.

L'ordinateur présente une différence capitale avec une calculatrice: il possède une unité de commande.

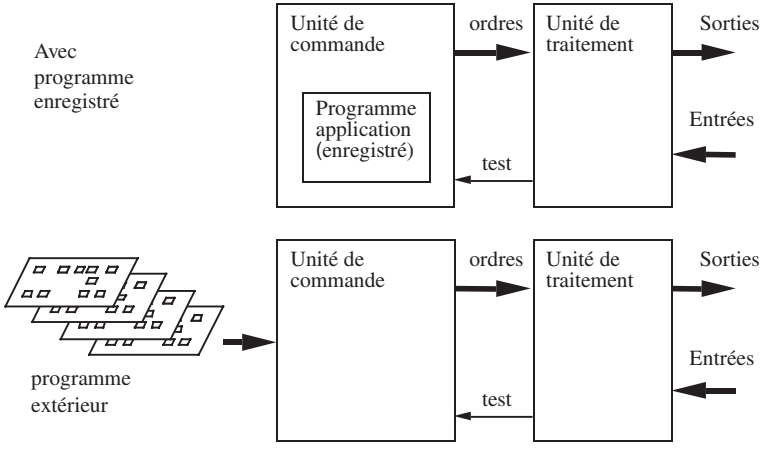


Fig. 4.1 Modèles d'ordinateurs avec et sans programme enregistré.

On peut considérer deux types d'unité de commande, selon qu'il y ait un programme enregistré ou non dans la machine (fig. 4.1). Lorsque ce n'est pas le cas, l'unité de commande est conceptuellement identique à celle de la machine de Babbage (chap. 2).

☞ Le programme peut être par exemple fourni à la machine par une suite de cartes perforées. Celles-ci sont alors lues les unes après les autres, avec pour conséquence la quasi-impossibilité de se brancher automatiquement à une autre partie du programme. Personne n'est en effet théoriquement là pour reprendre une partie des cartes déjà lues et les redonner à lire à la machine en cas de branchement en arrière.

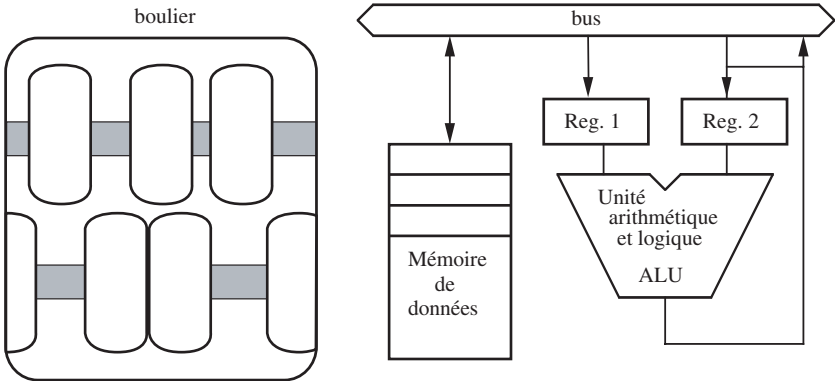


Fig. 4.2 Unité de traitement d'un ordinateur.

Quand l'unité de commande comporte le programme enregistré dans une mémoire à l'intérieur de la machine, il est possible de se brancher automatiquement à une autre partie de ce programme. Comme l'illustre la figure 4.3, le programme est mémorisé dans une mémoire à l'intérieur de la machine. Un registre pointeur de cette mémoire est généralement incrémenté pour lire l'instruction suivante. Mais ce registre peut aussi être chargé par une autre valeur, dite adresse de saut, permettant de se brancher à une autre partie du programme. Celle-ci est ordinairement fournie par une instruction du programme, dite instruction de branchement. Les autres instructions sont séquencées en plusieurs étapes (ou micro-instructions) afin d'être exécutées par l'unité de traitement.

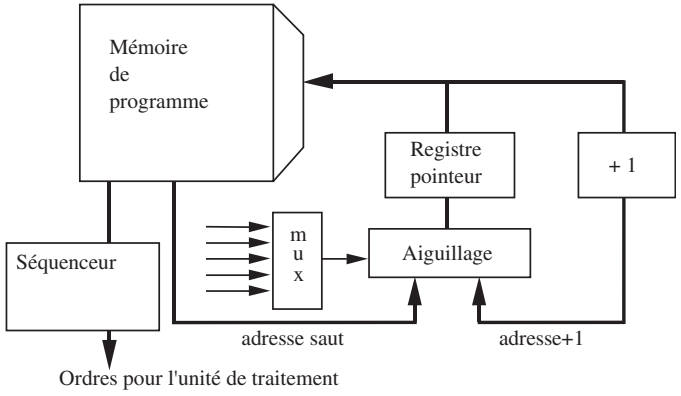


Fig. 4.3 Unité de commande avec programme enregistré.

La distinction entre programme enregistré ou non est fondamentale dans le concept même d'ordinateur. Seules les machines dotées de programmes enregistrés devraient être qualifiées d'«ordinateurs», les autres étant seulement des «calculateurs».

Le mot anglais «computer» ne permet bien entendu pas cette distinction, et c'est pourquoi on appelle *ordinateurs* les machines pourvues de programmes non enregistrés. Notons que les tous premiers ordinateurs ne sont en fait rien d'autre que des calculateurs dotés de programmes non enregistrés dans la machine…

Exécution de trois instructions dans un ordinateur

☞ Afin d'illustrer la manière dont procède un ordinateur pour exécuter des instructions, considérons un programme simple, dont le but est d'exécuter une addition de deux nombres logés dans une mémoire de données, et de conserver le résultat dans cette mémoire. Ce programme est le suivant:

```
CHARGE A, RAM(60);
ADDITION A, RAM(22);
SAUVE RAM(3), A;
```

La première instruction a pour objet d'aller chercher le nombre enregistré à l'adresse 60 de la mémoire de données, que l'on appele RAM. Ce nombre est alors stocké dans un registre A, appelé accumulateur. La deuxième instruction consiste elle aussi à aller chercher un nombre, mémorisé quant à lui à l'adresse 22. On additionne alors les deux nombres et le résultat est à nouveau enregistré dans l'accumulateur A. Le but de la troisième instruction est de sauver ce résultat dans une position de la mémoire de données RAM à l'adresse 3.

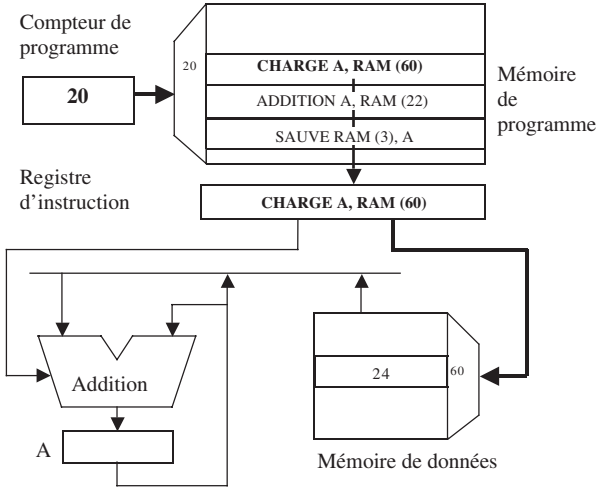


Fig. 4.4 Lecture de l'instruction CHARGE A, RAM(60).

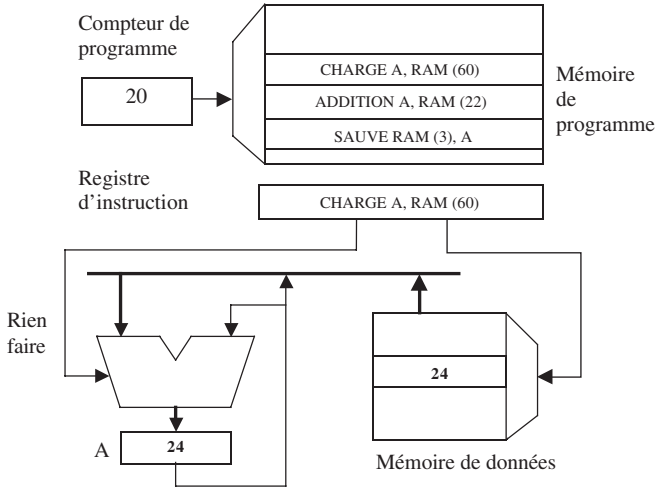


Fig. 4.5 Exécution de l'instruction CHARGE A, RAM(60).

La figure 4.4 illustre la lecture de la première instruction. Le compteur de programme comporte l'adresse 20 qui pointe l'instruction correspondante. Celle-ci est lue dans la mémoire de programme, avant d'être enregistrée dans un registre d'instruction placé à la sortie de cette mémoire. Un champ de cette instruction a la valeur 60, correspondant à l'adresse de la donnée qu'il faut aller chercher dans la mémoire de données.

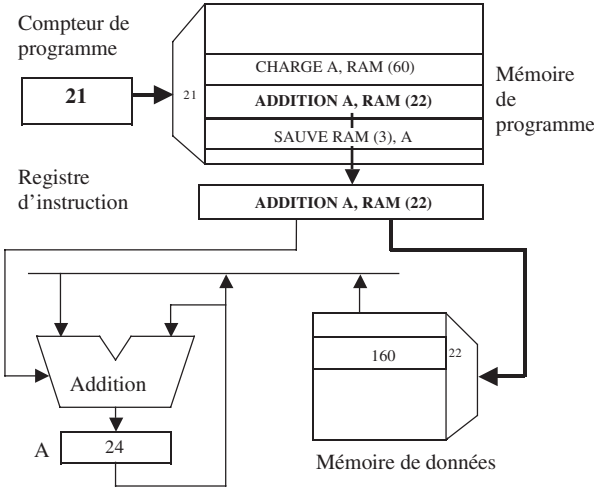


Fig. 4.6 Lecture de l'instruction ADDITION A, RAM(22).

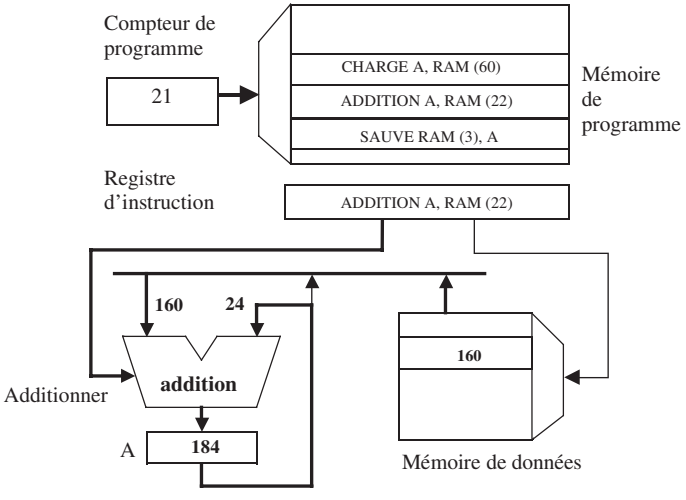


Fig. 4.7 Exécution de l'instruction ADDITION A, RAM(22).

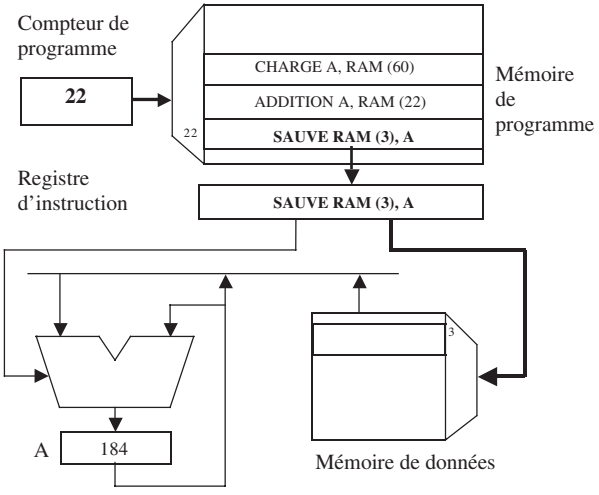


Fig. 4.8 Lecture de l'instruction SAUVE RAM(3), A.

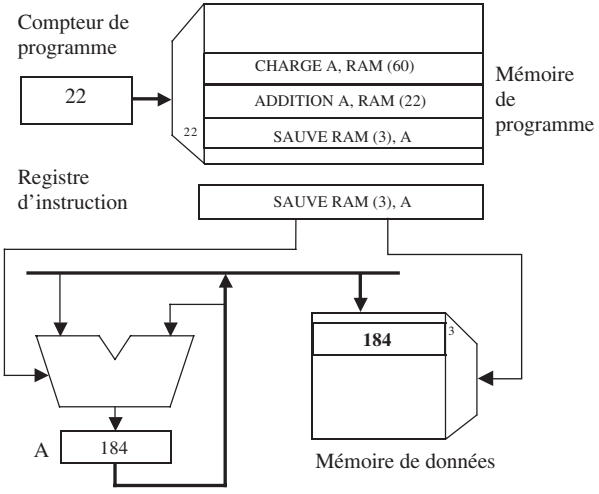


Fig. 4.9 Exécution de l'instruction SAUVE RAM(3), A.

La figure 4.5 représente l'exécution de cette première instruction. La donnée 24 logée à l'adresse 60 de la mémoire de données est lue et envoyée à l'unité de traitement. L'unité de calcul ne fait rien, sinon de laisser passer la donnée 24 pour qu'elle soit chargée dans l'accumulateur A.
La figure 4.6 est la lecture de la deuxième instruction, le compteur de programme ayant été augmenté de «1» pour pouvoir adresser l'instruction suivante. Celle-ci est à nouveau stoc-

kée dans le registre d'instruction et l'adresse 22 est envoyée à la mémoire de données pour y lire le deuxième opérande de l'addition.
La figure 4.7 décrit l'exécution de cette deuxième instruction. La donnée 160 est lue en mémoire de données et envoyée sur la première entrée de l'unité de traitement. La deuxième entrée est constituée par l'accumulateur A qui contient le nombre 24. L'instruction logée dans le registre d'instruction indique à l'unité de traitement qu'elle doit additionner les deux nombres à ses entrées et stocker le résultat 160 + 24 = 184 dans l'accumulateur A.
La figure 4.8 représente la lecture de la troisième instruction, le compteur de programme contenant alors 22. L'adresse de la position de la mémoire de données où il faudra mémoriser le résultat est envoyé à la mémoire de données.
Enfin, la figure 4.9 décrit l'exécution de cette troisième instruction. Le contenu de l'accumulateur A, soit le résultat 184, est envoyé à la mémoire de données et mémorisé à l'adresse 3.

Ordinateurs électromécaniques avec programmes non enregistrés

Il existe deux histoires parallèles de l'invention de l'ordinateur. L'une est celle de Konrad Zuse, l'autre celle des machines construites aux Etats-Unis.

Konrad Zuse

On considère aujourd'hui l'Allemand Konrad Zuse (1910-1995) comme l'inventeur de l'ordinateur, mis au point pour la première fois en 1941. C'est dans son appartement qu'il construisit dès 1936 un premier ordinateur entièrement mécanique [1] appelé Z1 qui utilisait le système binaire, puis apparaîtront successivement et jusqu'en 1945 les Z2, Z3 et Z4 électromécaniques. On considère que son ordinateur Z3 de 1941 fut le premier ordinateur complètement automatique, capable de lire un programme fourni sur bande perforée. Ce n'est qu'après la guerre que Konrad Zuse réalisa qu'il avait été le premier à mettre au point une telle machine.

Ne connaissant ni les travaux antérieurs réalisés sur la machine analytique de Babbage ni les calculatrices mécaniques avec des roues à dix positions, Zuse choisit le système binaire.

Une pièce est poussée ou tirée pour représenter le «0» ou le «1» rendant relativement facile la fabrication d'une mémoire mécanique. Zuse imagine également une porte logique mécanique qui peut ou non transmettre le mouvement d'une pièce à une autre selon la position d'autres pièces (fig. 4.10). Le programme est fourni sur un film 35 mm comportant des trous faits à la main. La mise au point de l'unité de calcul lui cause dans un premier temps énormément de difficultés.

Konrad Zuse utilise une représentation en virgule flottante (par exemple 26×10^3, 26 étant la mantisse et 3 l'exposant). Ni le Mark I, ni le ABC, ni l'ENIAC, que nous présentons plus loin, n'utilisaient la représentation en virgule flottante, plus complexe que celle en virgule fixe.

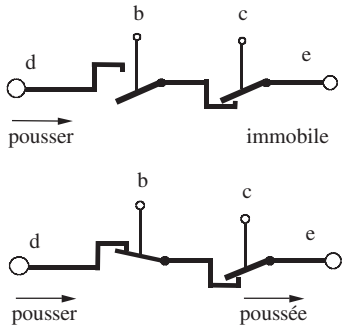


Fig. 4.10 Portes mécaniques binaires du Z1 de Zuse.

Comme Babbage avant lui, Zuse sépare complètement l'unité de traitement, la mémoire de données et l'unité de commande.

Il a en effet cette idée avant Von Neumann, et ne connaît pas à cette époque les travaux de Babbage sur la machine analytique. En 1939, Zuse termine le Z2, qui est la première machine utilisant des relais de téléphone électromécaniques «ouvert/fermé», parfaitement adaptés au système binaire choisi. Un de ses amis, Helmut Schreyer, tente de remplacer les relais par des tubes, mais abandonne à cause de la difficulté de se procurer un tel matériel, rare à l'époque. La mémoire du Z2 est constituée de la mémoire entièrement mécanique du Z1, mais l'unité de calcul comporte environ 1000 relais. Bien que fonctionnelle, cette machine est très peu fiable. Elle est cependant présentée à l'Institut de recherche aéronautique allemand, qui accepte de financer le développement d'une troisième machine, le Z3.

En 1941, Zuse construit le Z3, soit le premier véritable ordinateur. Contrairement à ses prédecesseurs, il fonctionne de manière indéfectible, au moyen d'un programme fourni par bande perforée.

La mémoire de 64 mots est elle aussi réalisée avec des relais, ce qui porte leur nombre à 2600 environ. C'est également une machine en virgule flottante avec 14 bits pour la mantisse et 7 bit pour l'exposant (et un bit de signe). L'ordinateur réalise 3 ou 4 additions par seconde. Avec un programme sur cartes perforées, il ne peut donc réaliser de branchements conditionnels. La figure 4.11 illustre la structure du Z3 et montre la séparation entre l'unité arithmétique et la mémoire de données, reliées entre elles par un bus de données de 22 bits (soit 22 fils). L'unité de contrôle permet de commander la lecture des instructions au travers d'un lecteur, d'exécuter les instructions dans l'unité arithmétique et de commander le clavier et l'affichage des résultats [14].

☞ Le jeu d'instructions du Z3 comporte 9 instructions différentes réparties en trois groupes (fig. 4.12). Elles sont extrêmement courtes, puisqu'elles ne comportent que 8 bits de large. Les instructions arithmétiques s'exécutent avec deux registres R1 et R2 avec résultat dans R1,

ces registres étant logés dans l'unité arithmétique illustrée à la figure 4.11. Il n'existe aucune instruction de branchement conditionnel; le Z3 est une machine à calculer qui exécute un programme de calcul sans pouvoir tester une condition [14].

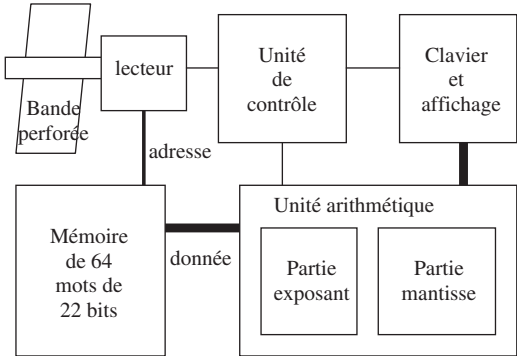


Fig. 4.11 Structure de l'ordinateur Z3 de Konrad Zuse.

Groupe	Instruction	Description	Code binaire
entrée/sortie	Lu	lire le clavier	01 11 0000
entrée/sortie	Ld	afficher	01 11 1000
lire la mémoire	Pr z	lire à l'adresse z	11 z6 z5 z4 z3 z2 z1
écrire la mémoire	Ps z	écrire à l'adresse z	10 z6 z5 z4 z3 z2 z1
arithmétique	Lm	multiplication	01 00 1000
arithmétique	Li	division	01 01 0000
arithmétique	Lw	racine carrée	01 01 1000
arithmétique	Ls1	addition	01 10 0000
arithmétique	Ls2	soustraction	01 10 1000

Fig. 4.12 Jeu d'instructions du Z3.

L'unité de commande du Z3 est formée de plusieurs roues correspondant à chaque instruction. Un bras conducteur d'électricité peut se déplacer de la gauche vers la droite pour faire successivement contact et ainsi activer les différents relais du Z3 [14].

C'est ainsi que toute une séquence d'ordres peut être donnée à l'unité de traitement pour, par exemple, exécuter une multiplication. Cette technique est très proche de la microprogrammation. La figure 4.13 illustre l'une des roues de l'unité de commande du Z3, similaire dans son principe au cylindre d'une boîte à musique.

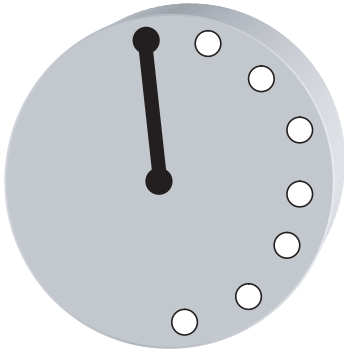


Fig. 4.13 Roue de contrôle du Z3.

Toutes ces machines furent détruites lors des raids aériens sur l'Allemagne. Mais en 1942, Konrad Zuse commence la conception du Z4, qui sera dévoilé au monde en 1947 avant d'être installée à l'EPFZ de Zurich en 1950. Zuse poursuivra ses travaux en Allemagne en construisant des ordinateurs entièrement électroniques tels que le Z22 (1950-1956). Ses recherches ne furent jamais financées par le régime nazi qui a probablement ignoré ces travaux fondateurs.

Georges Stibitz

En 1937, Stibitz (1904-1995), employé chez Bell Labs aux Etats-Unis, utilise des relais de téléphone pour construire, à titre personnel et durant ses week-ends, un additionneur binaire electromécanique. Deux ans plus tard, les Bell Labs décident d'utiliser cette idée pour construire une calculatrice (addition, soustraction, multiplication et division) de nombres complexes. Celle-ci comporte 450 relais et est reliée par lignes téléphoniques à trois terminaux, ce qui lui permet d'être utilisée par trois personnes à la fois, avec un simple mécanisme de priorité. D'autres machines sont construites durant la guerre sur le même principe, et employées pour effectuer des calculs balistiques et de pointage de canons. Ce ne sont donc que des calculateurs, utilisant des données entrées avec des bandes papier. Le code binaire utilisé est redondant, ce qui permet à la machine de détecter des erreurs, caractéristique non négligeable compte tenu du manque de fiabilité de ces machines. La plus grosse d'entre elles comporte 9000 relais et pèse près de 10 tonnes.

Howard Aiken

A partir de 1939, Howard Aiken (1900-1973) construit, à l'Université de Harvard et en collaboration avec IBM et son patron T. J. Watson, un ordinateur électromécanique, le Mark I. Il semble connaître la machine de Babbage et s'en inspire. Le Mark

I se compose de 750 000 éléments, pèse 5 tonnes, lit et exécute une à une les instructions préalablement enregistrées sur des bandes papier. Il est aussi équipé d'autres lecteurs de bandes perforées pour introduire des données ou des tables d'informations. Le Mark I ne peut se brancher immédiatement à une autre partie du programme sans intervention manuelle, défaut récurrent dans les machines non dotées de programme enregistré. Le format de l'instruction est représenté à la figure 4.14, OP étant l'opération, A1 le registre du 1er opérande, et A2 à la fois le registre du 2e opérande et celui du résultat [5].

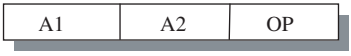


Fig. 4.14 Format de l'instruction du Harvard Mark I.

A l'instar de la machine de Babbage, le Mark I travaille en base 10. La mémoire de données est constituée de roues susceptibles d'occuper dix positions pour représenter un chiffre décimal, et un système de contacts permet de «lire» l'état des roues.

Ces roues peuvent avancer, constituant ainsi l'organe de calcul de la machine, sous forme de compteurs. La mémoire de données est composée de 72 roues, ou registres mécaniques, qui ont l'avantage de ne pas perdre leur information lors de lectures, comme cela se produisait avec la machine analytique de Babbage.

Jusqu'en 1962, Aiken est considéré comme l'inventeur de l'ordinateur, avant qu'il n'admette que Zuse l'a devancé de quelques années. Le Mark I fut terminé en 1944. Aiken construisit ensuite le Mark II avec des relais, puis les Mark III et IV avec des tubes à vide, ces derniers étant alors de véritables machines électroniques.

Ordinateurs électroniques avec programmes non enregistrés

Les ordinateurs présentés jusqu'ici sont des machines électromécaniques, réalisées avec des relais. Les ordinateurs électroniques que nous utilisons aujourd'hui n'apparaissent que récemment, juste avant la Seconde Guerre mondiale.

Les premiers dispositifs électroniques qui voient le jour sont des tubes à vide, et non des transistors. Les tubes électroniques sont utilisés par Stibitz en 1937 pour réaliser sa calculatrice, et Zuse et Schreyer envisagent en 1939 de les employer pour la construction du Z2.

John Atanasoff

L'idée d'ordinateur électronique nait aux USA avec John Atanasoff (1903-1995), de l'Université de Iowa. En 1935, il tente de fabriquer un premier modèle basé sur le système binaire, mais n'y parvient pas.

En 1937-38, Atanasoff se trouve confronté à la résolution de longs calculs, et décide alors de construire sa propre machine. Assisté de Clifford Berry, il entame en 1939 la construction de l'ABC (Atanasoff-Berry Computer) en employant des tubes à vide et comprend rapidement que le code binaire est le plus adapté au fonctionnement d'une machine électronique. Les plans de la machine sont prêts cette même année. L'additionneur-soustracteur en binaire comporte 210 tubes à vide, et le contrôle de la machine en exige encore 300 autres. La mémoire de données de 30 nombres de 50 digits est constituée de capacités montées sur deux cylindres rotatifs, et doit être rafraîchie à chaque tour du cylindre.

Cette machine ne peut cependant qu'additionner et soustraire. La lecture des données par cartes perforées est très lente, et l'ordinateur ne fonctionne qu'à 60 Hz (60 étapes d'exécution par seconde). La vitesse potentielle d'exécution des instructions offerte par l'électronique est donc loin d'être utilisée. Mais en 1942, alors que l'ABC est presque terminé, Atanasoff est appelé au Naval Ordnance Laboratory pour participer à l'effort de guerre américain. Il ne reprendra jamais la construction de l'ordinateur qui servira toutefois de modèle pour la construction de l'ENIAC.

Bien qu'inachevée, la machine de Atanasoff est considérée comme le premier ordinateur binaire électronique. En 1971, une cour américaine décide que le brevet sur l'ENIAC avait été antériorisé par les idées de Atanasoff, et que celui-ci était le véritable inventeur de l'ordinateur [2]. John Atanasoff décède en 1995, la même année que Zuse, Eckert et Stibitz.

L'ENIAC

Considéré à tort comme le premier ordinateur électronique, l'ENIAC (Electronic Numerical Integrator And Computer) est construit à l'Université de Pennsylvanie entre 1943 et 1944, et voit le jour en novembre 1945. C'est le dernier très gros ordinateur avec programme non enregistré dans la machine.

Inventé par John Mauchly (1907-1980) et John-Presper Eckert (1919-1995), l'ENIAC doit pouvoir calculer rapidement des tables balistiques, comme le souhaitent alors le Ballistic Research Laboratory et son représentant, le lieutenant Goldstine, commanditaires de l'appareil.

De par sa nature, et notamment l'absence de branchements conditionnels, l'ENIAC est plus proche d'une grosse calculatrice que d'un ordinateur. Il pèse 30 tonnes, consomme 150 000 Watts, comporte 18 000 tubes à vide, et sa programmation nécessite de re-connecter les différentes unités de manière adéquate. Il travaille à la vitesse de l'électronique, soit 200 000 Hz ou étapes par seconde, c'est-à-dire beaucoup plus rapidement que les ordinateurs électromécaniques. Impossible par conséquent d'employer un programme sur cartes perforées avec une telle machine, car le processus serait beaucoup trop lent par rapport à la vitesse offerte par l'électronique.

☞ La figure 4.15 représente la structure de l'ENIAC, équipé de 20 accumulateurs de dix digits décimaux [5]. Programmé manuellement en établissant les connexions adéquates, par

exemple entre A1 et A2 si l'on désire additionner le contenu de ces deux accumulateurs, l'entrée des données s'effectue par cartes perforées, et la table de fonctions comporte des constantes.

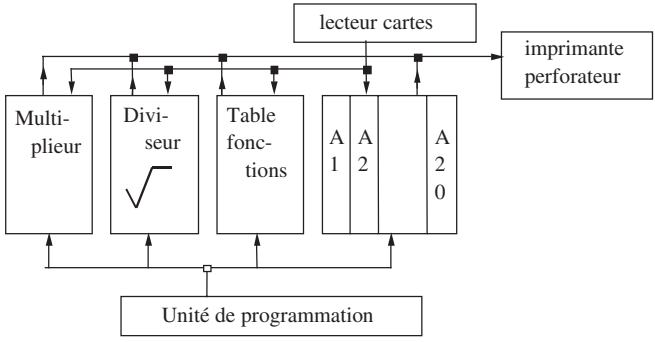


Fig. 4.15 Structure de l'ENIAC.

Bien qu'électronique, la structure de l'ENIAC copie celle des ordinateurs électromécaniques. Les roues ne sont pas mécaniques, mais consistent en 10 tubes à vide en guise de compteur à 10 états, dont un seul à la fois est actif.

☞ La figure 4.16 offre un aperçu imagé de ce fonctionnement: pour ajouter 2 à un compteur dans l'état actif 4, on envoie deux impulsions qui font avancer le compteur à l'état 6, qui devient alors actif. Si on passe de 9 à 0, le report est transmis sous forme d'une impulsion au compteur représentant le digit d'ordre supérieur.

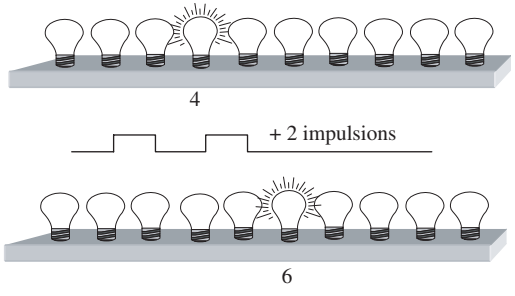


Fig. 4.16 Roue électronique de l'ENIAC.

L'appareil comporte 20 de ces accumulateurs dotés chacun de dix digits. La machine peut aussi multiplier et diviser.

Le choix de la base 10 plutôt que de la base 2 peut paraître étrange – bien que Mauchly ait rencontré Atanasoff à l'Université de Iowa – car Leibnitz, Zuse et Atanasoff ont montré que la base 2 est plus avantageuse pour les machines électro-

niques. Ceci démontre bien l'influence que le Mark I de Aiken, et par extension la machine analytique de Babbage, exercent sur les esprits des concepteurs de l'ENIAC.

L'unité de commande de l'ENIAC contenant le programme est elle aussi électronique et consiste en plusieurs compteurs produisant une séquence d'impulsions de commande. Chaque compteur réalise une partie du programme, en fait une boucle (ou un sous-programme) qui peut être répétée plusieurs fois. Pour qu'un compteur particulier réalise la partie désirée d'un programme, il faut connecter ses sorties aux lignes de commande des unités de calcul, câblage qui peut être long, et tient lieu de programmation de l'appareil.

Mais l'ENIAC est fragile, et sujet à une panne tous les trois jours en moyenne. Il ne doit pas être arrêté durant la nuit, car tout changement de température, notamment ceux survenant lors du redémarrage, risque de détruire les tubes. En 1949, G. Reitwiesner l'utilise pour calculer les 2 037 décimales du nombre π, calcul qui nécessite près de 70 heures. L'ENIAC fonctionne durant dix ans, jusqu'en octobre 1952. Il est modifié pour utiliser les tables de constantes et y loger des instructions, afin d'équiper la machine d'un petit programme enregistré.

Après Ada Lovelace, c'est encore aux femmes que l'on doit les programmes de l'ENIAC…

Beaucoup de femmes furent engagées pendant la guerre pour calculer manuellement les tables de balistique, et ceci explique peut-être pourquoi certaines d'entre elles devinrent programmeuses. Adele Goldstine écrivit un manuel de programmation de l'ENIAC, et Betty Holberton reçut en 1997, le jour de ses 80 ans, le prix Augusta Ada Lovelace (prix de la National Association for Women in Computing). Kathleen Mauchly demeure elle aussi associée à l'épopée de l'ENIAC.

NOM	TYPE	Année	Inventeur	Mémoire	Remarques
Machine Analytique	mécanique	1833-....	Babbage	1000×50 digits	cartes perforées
Z1	mécanique	1936	Zuse	–	binaire, flottant
Z3	relais	1939-1941	Zuse	64×22 bits	binaire, flottant
Bell Labs	relais	1938-1940	Stibitz	15×7 digits	5 lecteurs bande
Mark I	électro-méc.	1939-1944	Aiken	72×23 digits	lecteur bande
SSEC	relais	1945-1948	Hamilton	163×19 digits	38 lecteurs bande
Iowa	électronique	1939-....	Atanasoff	64×50 bits	équations algébriques
ENIAC	électronique	1943-1946	Eckert, Mauchly	20×19 digits	program. câblée

Fig. 4.17 Classement des premiers ordinateurs.

Notons que l'histoire a probablement retenu l'ENIAC comme premier ordinateur électronique non pas en raison des qualités de la machine elle-même, mais plutôt à cause des personnalités qui ont œuvré à sa conception (fig. 4.17). Outre les inventeurs,

apparaît en particulier le nom de John Von Neumann, qui conceptualisera plus tard de manière extrêmement claire la structure des ordinateurs.

Ordinateurs électroniques avec programmes enregistrés

John Von Neumann (1903-1957)

Tant que les instructions d'un ordinateur doivent être fournies de l'extérieur, celui-ci demeure un outil très limité. Rendre possible le test de conditions et l'exécution de boucles demande au préalable d'imaginer un moyen pour mémoriser le programme à l'intérieur de la machine. L'utilisation de bandes papier devient impossible, compte tenu de la vitesse de calcul des machines électroniques.

Une rencontre fortuite mais décisive à la gare d'Aberdeen en janvier 1944 entre Von Neumann et le lieutenant Goldstine, qui suit pour l'armée US le développement de l'ENIAC, incite Von Neumann à venir visiter la Moore School où est construit l'ENIAC. La machine est alors presque terminée, et Von Neumann s'intéresse surtout à l'organisation logique de l'ordinateur. Devenu consultant, il discute des améliorations que l'on pourrait apporter à l'architecture de futurs ordinateurs électroniques, et commence à travailler sur le projet EDVAC (Electronic Discrete VAriable Computer) en compagnie des créateurs de l'ENIAC, Mauchly et Eckert. Après de nombreuses discussions entre eux au sujet des limitations de l'ENIAC, Von Neumann publie un rapport préliminaire de quelques pages daté du 30 juin 1945 dans lequel il décrit l'architecture d'un ordinateur avec programme enregistré dans la mémoire même de la machine (fig. 4.18). A ce titre, il est considéré comme le père des architectures d'ordinateurs avec programmes enregistrés, architectures encore en usage aujourd'hui. Bien que Von Neumann signe seul son rapport, il semble que l'idée est celle de Mauchly et Eckert, ce dernier ayant pensé à mémoriser les instructions d'un ordinateur sur un disque magnétique à l'intérieur de la machine dès fin 1943 ou début 1944 déjà. Le rapport de Von Neumann ne comporte toutefois pas seulement l'idée de programme enregistré, connue depuis Babbage et Aiken. La figure 4.18 représente une architecture à programme enregistré appelée Harvard, en référence à la machine Mark I construite à Harvard par Aiken, et qui comporte deux mémoires différentes pour le programme et les données. L'un des inconvénients de cette architecture est d'exiger une répartition fixe entre le nombre d'instructions et le nombre de données possible.

Mais Von Neumann ne se limite pas à simplement juxtaposer une unité de commande avec un programme enregistré et une unité de traitement. Il a surtout l'idée d'une architecture ne comportant qu'une seule mémoire, incluant à la fois les instructions et les données (fig. 4.19). Outre une remarquable simplicité liée au nombre réduit de mémoires, l'architecture, dite «de Von Neumann» offre une grande souplesse, les programmes pouvant comporter beaucoup d'instructions et peu de données, ou inversement. Elle permet aussi de charger un autre programme ou une partie du programme dans la mémoire, comme on le ferait pour des données.

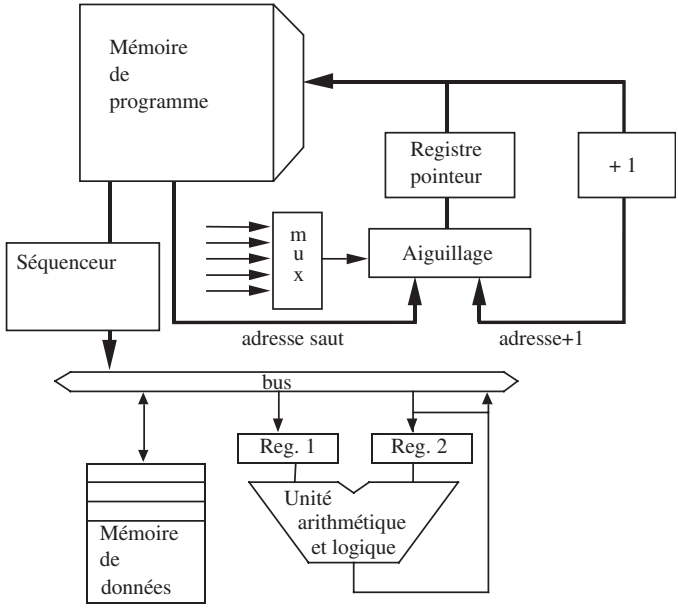


Fig. 4.18 Ordinateur avec programme enregistré et deux mémoires (Harvard).

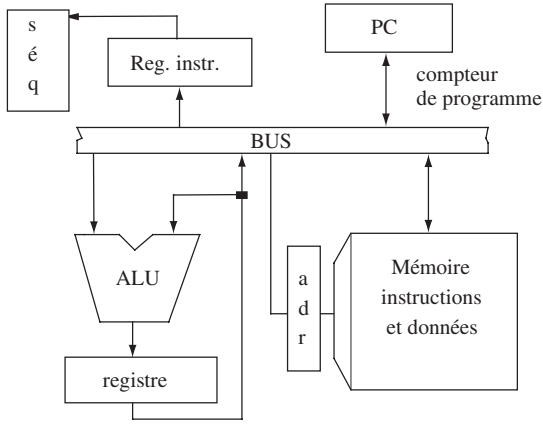


Fig. 4.19 Architecture de Von Neumann avec une seule mémoire.

Si les architectures des ordinateurs modernes (Von Neumann ou Harvard) naissent avec les plans de l'EDVAC, la construction de la machine se déroule de manière très laborieuse en raison de la mésentente entre Von Neumann et Mauchly/Eckert. L'EDVAC ne voit le jour qu'en 1952, alors que d'autres grands ordinateurs avec pro-

grammes enregistrés, inspirés des travaux publiés par Von Neumann, apparaissent entre-temps, comme le SSEC d'IBM en 1948, l'EDSAC en 1949, le BINAC en 1949 ou l'UNIVAC en 1951.

Les répertoires d'instructions

Installer le programme dans la mémoire où résident aussi les données implique que les données et les instructions aient le même format, soit le même nombre de bits. Cela n'est pas sans conséquence pour les instructions, car si la donnée est de 16-bits ou de 32-bits, l'instruction doit en principe être de même. En fait, certaines instructions peuvent comporter plusieurs mots mémoire, par exemple 2 ou 3 mots de 16-bits.

Comment donner le sens voulu à une instruction formée de 16 ou 32 bits? Combien y aura-t-il d'instructions différentes? Comment écrit-on un programme pour une machine donnée ? Répondre à ces questions, c'est en fait concevoir un répertoire ou un jeu d'instructions. Cette étape est très importante: elle fait le lien entre le matériel et le logiciel.

☞ La figure 4.20 représente une instruction avec trois champs. Le premier champ ADD informe l'unité de calcul qu'elle doit sélectionner l'opération d'addition. Les deux champs suivants sont dits «d'adresse», et sélectionnent dans la mémoire les deux opérandes de l'addition. Dans cet exemple, le champ adr1 est par ailleurs aussi un champ de destination, le résultat étant sauvé à cette adresse.

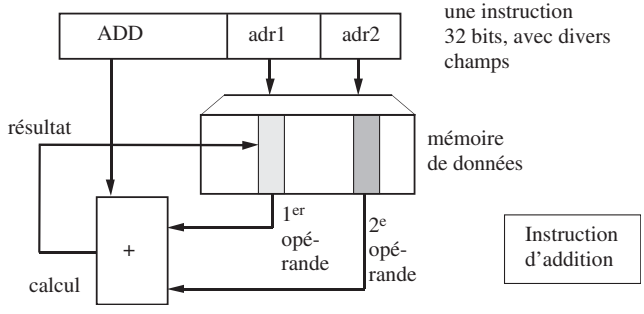


Fig. 4.20 Champs d'une instruction et leurs effets sur la machine.

Les mémoires

Le grand problème d'un ordinateur avec programme enregistré était la mémoire. Il n'était pas possible de la réaliser avec des relais ou d'autres dispositifs mécaniques (comme les mémoires de Zuse) qui étaient trop lents.

Le type de mémoire le plus largement utilisé est la ligne à retard, constituée d'un tube de mercure dans lequel on propage une onde acoustique.

☞ Pour écrire un bit, on envoie une impulsion électrique à l'entrée du dispositif, qui le convertit en onde acoustique par un quartz. Le bit se propage dans la ligne et relu en sortie par un autre quartz afin d'obtenir à nouveau une impulsion électrique. Une ligne à retard comporte généralement 1000 bits, et doit être constamment écrite et lue pour ne pas perdre l'information (fig. 4.21).

Ce dispositif est très sensible à la température, laquelle doit demeurer fixe afin de conserver un délai constant et déterminé afin d'identifier la nature des bits lus en sortie. Ce type de mémoire fut utilisé pour l'EDVAC, l'EDSAC et l'UNIVAC.

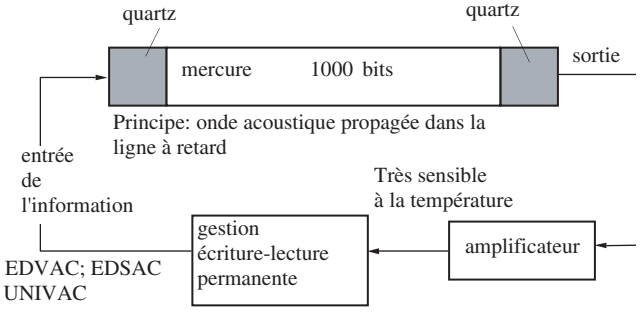


Fig. 4.21 Mémoire avec une ligne à retard à mercure.

Le problème posé par ce type de mémoire est la circulation continue des instructions. Après avoir lu une instruction, l'ordinateur doit l'exécuter, alors même que les autres instructions continuent de passer par la tête de lecture. Un tour complet est donc nécessaire pour lire et exécuter l'instruction suivante [6]. Pour éviter cela, on peut inclure dans le format de l'instruction l'adresse de la suivante, et les mémoriser dans la mémoire de manière espacée, et non pas consécutive.

Un deuxième type de mémoire est un tube CRT (Cathode Ray Tube) reposant sur le principe qu'un faisceau d'électrons arrivant sur une surface de phosphore y laisse une tache lumineuse pendant $2/10^{e}$ de seconde (fig. 4.22).

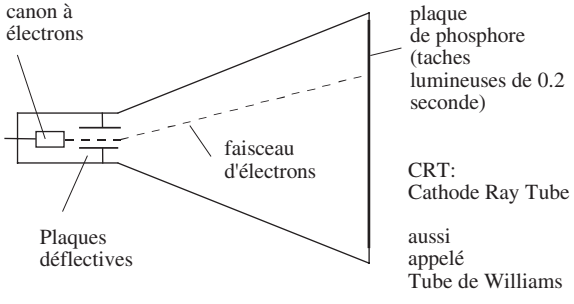


Fig. 4.22 Mémoire avec un tube CRT.

☞ Son fonctionnement est le suivant: un faisceau d'électrons est dirigé sur chaque position d'un bit d'une plaque de phosphore, en écrivant et lisant la mémoire au moins 5 fois par seconde. La lecture est faite en envoyant un nouveau faisceau d'électrons, dit «de lecture», sur la position d'un bit et le courant recueilli diffère selon que le bit est à «1» ou à «0». Dès que le bit est lu, un nouveau faisceau d'électrons d'écriture vient rafraîchir la mémoire.

Un troisième type de mémoire est le tambour magnétique, gros cylindre sur lequel est déposé un matériel magnétique, et dont le principe est identique à celui des bandes magnétiques modernes. Desservis par une vitesse d'écriture et de lecture peu élevée, ils sont surtout utilisés comme mémoires lentes (comme les disquettes et les disques durs actuels).

Les machines Colossus de Alan Turing

Von Neumann n'est pas le seul inspirateur des ordinateurs modernes. L'Anglais Alan Turing pourrait même être le premier à avoir construit un ordinateur électronique.

Alan Turing (1912-1954) est un homme très solitaire et profondément anticonformiste [7]. En 1935, il décrit «la machine de Turing», appareil universel équipé d'un programme enregistré, capable d'exécuter n'importe quel algorithme. Cette machine n'a jamais été construite car Turing s'intéresse plus aux concepts abstraits de la structure logique d'un ordinateur qu'à sa réalisation concrète. Le principe de l'appareil était le suivant: un problème pouvant être programmé sur cette machine est décidable, c'est-à-dire qu'il ne reste pas sans solution; autrement dit un problème a une solution ou non (chap. 5).

De 1939 à 1942, à Bletchey Park, Alan Turing [18] conçoit la machine mécanique «Bombe», utilisée pour décrypter les messages allemands de la machine Enigma [16]. Il devient ensuite consultant pour la conception des machines anglaises Colossus, construites par Max Newman, et dont l'objectif est le décryptage du «chiffre de Lorentz», beaucoup plus complexe que celui des messages de la machine Enigma (fig. 4.23) [16]. Colossus I est conçue en 1940 à base de relais. Colossus II, mise au point en 1943 par l'ingénieur Tommy Flowers et livrée à Bletchey Park le 6 décembre de cette même année [16], est équipée de tubes électroniques. Dix exemplaires de la machine sont construits.

Colossus II est demeuré secret jusqu'au milieu des années 1970. On ne sait que depuis peu que la machine a fonctionné avant l'ENIAC [8], et qu'elle constitue donc le tout premier ordinateur électronique.

Le Colossus II comporte 2500 tubes, et les instructions sont lues sur des bandes perforées et stockées dans des mémoires tampons (réalisées avec des tubes). Le programme est ainsi partiellement mémorisé dans la machine [7]. L'idée de mémoire temporaire («mémoire cache») date donc des premiers ordinateurs.

Une machine Colossus a été reconstruite récemment à Bletchey Park.

Fig. 4.23 Machine Enigma (Musée de Paderborn, Allemagne).

Le SSEC d'IBM

Le nom de la compagnie IBM est étroitement associé aux ordinateurs. Fondée en 1911 par Hollerith, l'inventeur du totalisateur électrique utilisé pour le dépouillement du recensement de 1890, la compagnie prend son nom définitif en 1924 (chap. 2). Son ascension débute dès la fin de la guerre.

En 1947, IBM construit un ordinateur hybride inspiré du Harvard Mark I de Aiken, le SSEC (Selective Sequence Electronic Computer), doté de 13 000 tubes à vide pour la partie arithmétique, 8 registres rapides et 23 000 relais pour la partie de contrôle, et 150 registres plus lents. Inauguré en janvier 1948, il a la particularité d'être à la fois le dernier et le plus complexe des monstres électromécaniques jamais construits (doté par ailleurs d'une partie électronique très importante). Des tubes à vides électroniques constituent les parties rapides (l'arithmétique), alors que des relais moins chers sont utilisés pour la partie lente. Le SSEC peut exécuter 50 instructions par seconde.

Les 13 000 tubes de la partie électronique sont utilisés plus efficacement que ceux de l'ENIAC qui nécessite 10 tubes pour mémoriser un digit. Dans le SSEC, le codage des 10 digits 0 à 9 est réalisé par un code binaire dit BCD (Binary Coded Decimal) qui n'exige que 4 tubes à vide. Bien qu'équipé de moins de tubes, le SSEC parvient à la même puissance de calcul que l'ENIAC.

Comme pour les autres ordinateurs électromécaniques, le programme du SSEC est écrit sur bande papier perforée. 66 lecteurs équipent la machine, et lui offrent une grande flexibilité de choix des programmes. Mais la principale particularité de cette

machine est qu'elle est dotée de quelques registres internes pouvant contenir des instructions: c'est le concept de programme enregistré. Les instructions sont lues par le lecteur de bandes, et mémorisées dans ces registres. Il est alors possible de re-exécuter le programme, dans le cas d'une boucle par exemple, ou de se brancher à une autre partie du programme, voire même de sauter quelques instructions selon des résultats intermédiaires.

Premier ordinateur avec programme enregistré mis en service, le SSEC est utilisé par les services commerciaux d'IBM, puis loué par le gouvernement des Etats-Unis pour la Commission atomique, avant d'être définitivement abandonné en 1952.

La machine «Baby» de Manchester

Dès la publication du rapport de Von Neumann et des séminaires de la Moore School, des projets similaires voient le jour en Angleterre, en particulier à Manchester et à Cambridge.

Une première machine est construite à Manchester par Tom Kilburn, en collaboration avec Freddie Williams qui est le père des tubes CRT dits tubes «Williams» (fig. 4.22). Son premier but est de prouver que les tubes CRT peuvent être utilisés comme mémoires d'ordinateurs. L'architecture choisie est de type série, les bits d'une donnée sont traités un à un. Surnommé «Baby», cet ordinateur fonctionne pour la première fois le 21 juin 1948, représentant du même coup le premier ordinateur électronique à programme enregistré fonctionnel (fig. 4.24). Ses capacités sont toutefois très limitées par la taille réduite de la mémoire, constituée d'un tube CRT (fig. 4.22), et dotée d'une capacité de 32 mots de 32 bits. Le premier programme de démonstration ne comporte donc que 17 instructions et 15 positions mémoire pour les données. C'est insuffisant pour de vrais programmes

☞ Le répertoire d'instructions de cette machine est constitué de 7 instructions différentes de 32 bits, dont 16 bits inutilisés (fig. 4.25):

1) accumulateur ← valeur inversée de la mémoire(adresse);
2) accumulateur ← accumulateur – mémoire(adresse);
3) mémoire(adresse) ← accumulateur;
4) si accumulateur < 0, sauter la prochaine instruction;
5) pointeur instruction ← mémoire(adresse);
6) pointeur instruction ← pointeur instruction + mémoire(adresse)
7) stop;

Les capacités de la machine sont par la suite étendues grâce à l'adjonction d'une mémoire à tambour magnétique de 1024 mots et deux tubes CRT de 128 mots. Le programme du tambour est chargé dans les tubes CRT lors de l'exécution, et la machine comporte 50 instructions différentes. Mise au point en avril 1949, elle est commercialisée par Ferranti sous le nom de Ferranti Mark I (créant une confusion avec la machine Mark I de Aiken), au prix d'environ 500 000 $ l'unité.

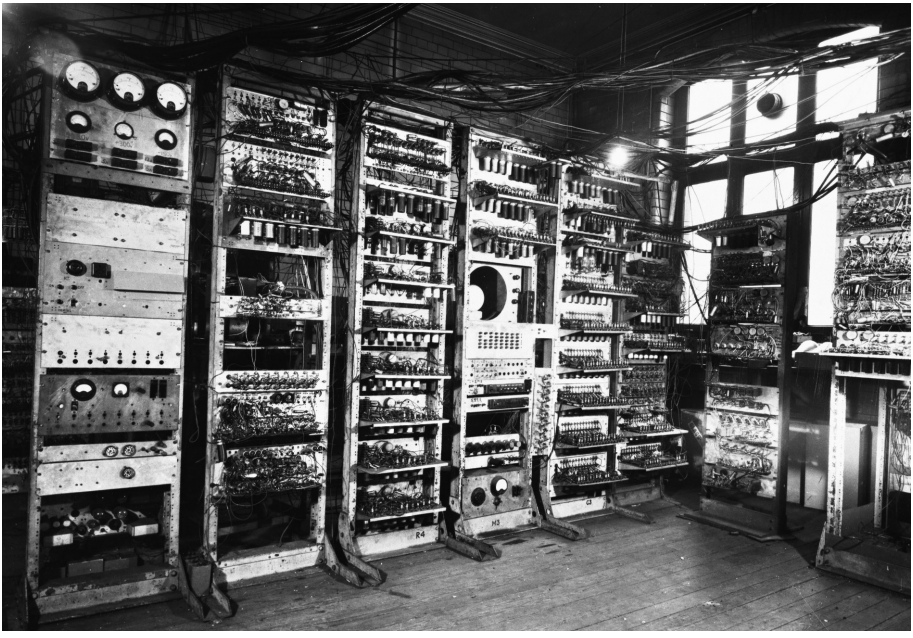


Fig. 4.24 L'ordinateur «Baby» de Manchester (1948).
(Département de Computer Science, Université de Manchester, Royaume Uni)

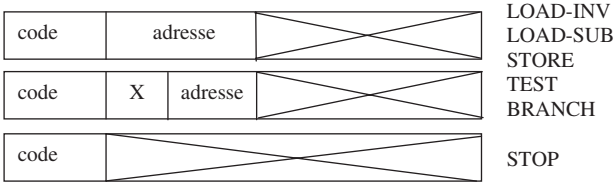


Fig. 4.25 Formats d'instructions de la machine de Manchester.

La machine EDSAC de Cambridge

La machine EDSAC (Electronic Delay Storage Automatic Calculator) fut construite par le professeur Wilkes sur la base du rapport de Von Neumann. Il choisit une mémoire constituée d'une ligne à retard à mercure (fig. 4.21) avec 16 tubes de 32 mots de 17 bits. Les données comportent 10 digits, soit un mot de 36 bits. La machine comprenant 18 instructions différentes de 17-bit (fig. 4.26), on peut donc loger deux instructions sur un mot de 36 bits. Il n'y a qu'une seule adresse d'un opérande dans l'instruction, et la machine travaille avec un accumulateur. Une instruction d'addition a par exemple la forme:

accumulateur <-- accumulateur + opérande (adresse)

Comportant 4000 tubes, l'EDSAC est mise au point en mai 1949. Les machines LEO, semblables à l'EDSAC sont les premiers ordinateurs à programme enregistré utilisés pour des applications commerciales.

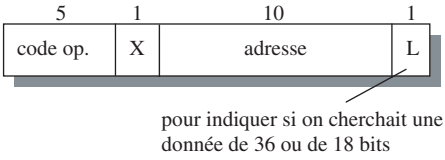


Fig. 4.26 Format d'une instruction 17 bits de l'EDSAC.

Difficultés pour l'EDVAC

Alors que l'ENIAC est terminé et que les plans de l'EDVAC sont en cours d'élaboration, la Moore School se trouve confrontée à divers problèmes lors de la première réalisation de cet ordinateur électronique à programme enregistré.

Le conflit qui éclate entre Von Neumann et ses deux collègues Mauchly et Eckert a plusieurs origines. D'abord, Von Neumann s'arroge la paternité des ordinateurs à programmes enregistrés en signant seul le fameux rapport «First Draft of a Report on the EDVAC», alors même que c'est Mauchly et Eckert qui ont eu les premiers l'idée d'un tel ordinateur. Travaillant pour l'armée, ces deux derniers sont tenus au secret, alors que Von Neumann, employé comme consultant, a toute liberté de parole. Enfin, ils ne peuvent accepter que Von Neumann utilise toute la connaissance de l'EDVAC afin de mettre au point la machine IAS pour le compte de RCA.

Un autre conflit apparaît bientôt, opposant Mauchly et Eckert à la Moore School à propos des brevets sur l'ENIAC et l'EDVAC dont les deux parties revendiquent les droits. Les juristes concluent à l'impossibilité de délivrer un brevet puisqu'il existe déjà le rapport de Von Neumann. Après une enquête de la Moore School pour déterminer comment l'argent avait été utilisé pour l'ENIAC et l'EDVAC, Mauchly et Eckert décident, à l'instar de Von Neumann, de quitter l'établissement. Plus personne ne poursuit alors de travaux sur l'EDVAC.

L'EDVAC est toutefois construit par une nouvelle équipe composée de Sharpless et Lukoff. La mémoire utilisée est une ligne à retard à mercure (fig. 4.21) et l'architecture de type série. La machine comporte deux unités de calcul en parallèle afin de comparer les résultats et de détecter d'éventuelles erreurs. Elle fonctionne à 1 MHz (un million de pulsations par seconde), et comprend 12 instructions différentes de 44 bits, dont 4 adresses de 10 bits chacune pour le premier et le second opérande, la destination du résultat ainsi que l'adresse de l'instruction suivante. Voici un exemple d'instruction (fig. 4.27):

destination(A3) <-- opérande (A1) + opérande (A2) avec OP = «+»
et se brancher à A4 pour la prochaine instruction à exécuter;

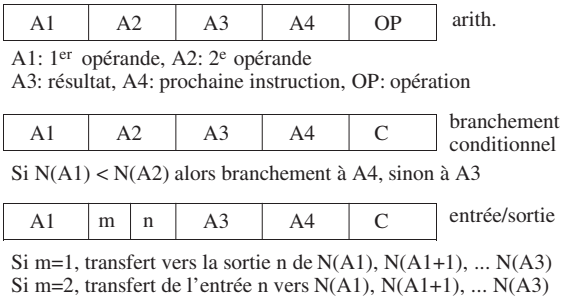


Fig. 4.27 Répertoire d'instructions de l'EDVAC.

Dotée d'un format d'instruction composé de nombreux champs, l'EDVAC possède une largeur d'instruction importante de 44 bits, contre 16 pour le Baby et 17 pour l'EDSAC.

☞ L'instruction de l'EDVAC est très large car la machine comporte trois adresses (A1, A2 et A3) de 10 bits chacune pour les deux opérandes et le résultat [5]. On peut donc adresser 2^{10} = 1024 mots. L'utilisation d'une mémoire à ligne de retard oblige chaque instruction à comporter une 4e adresse pour l'instruction suivante, l'ensemble composant une instruction de 44 bits. La figure 4.27 représente une instruction de branchement conditionnel basée sur la comparaison de deux données adressées par A1 et A2. L'instruction comporte les deux adresses A3 et A4 déterminant la prochaine instruction. La dernière instruction est celle d'entrée/sortie, l'adresse n distinguant le lecteur de bande, le perforateur ou l'imprimante qui doit être utilisé.

Une instruction large permet d'améliorer les performances de la machine, car on adresse en même temps les deux opérandes et le résultat plutôt que de le faire en séquence en utilisant un accumulateur (fig. 4.4 à 4.9). L'inconvénient majeur est toutefois la taille importante de la mémoire.

L'équipe de recherche change une nouvelle fois en été 1947 et l'EDVAC est finalement terminée en 1952, soit bien longtemps après toutes les autres machines inspirées du rapport de Von Neumann sur l'EDVAC. Elle comporte 3500 tubes à vide, et fonctionne jusqu'en 1962.

La machine IAS de Princeton

Von Neumann décide avec Goldstine de construire une machine à Princeton… il propose alors à Eckert de le rejoindre, mais ce dernier refuse.

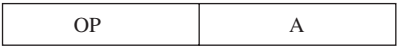
L'architecture de l'IAS est repensée par rapport à l'EDVAC. Une architecture parallèle traite les 40 bits d'une donnée en même temps (et non en série), comporte 4000 mots de 40 bits en mémoire, et fonctionne à 1 MHz (un million de pulsations par seconde). Les différentes unités de la machine travaillent à leurs propres rythmes et doivent signaler lorsqu'elles ont terminé leurs tâches; une telle architecture, dite asynchrone, n'avait jamais été construite auparavant. Seuls des microprocesseurs asyn-

chrones expérimentaux ont d'ailleurs été construits à ce jour, et aucun n'est encore disponible sur le marché.

Un ordinateur traitant les bits d'une donnée en parallèle demande une mémoire capable de lire tous les bits d'une donnée en même temps. L'équipe de Von Neumann voulut utiliser une mémoire construite par RCA, le Selectron, mais de graves difficultés techniques ne permirent pas à celle-ci de voir le jour. Le tambour magnétique s'avérera quant à lui trop lent pour un tel usage. La mémoire de l'IAS fut finalement réalisée, non sans peine, à partir de 40 tubes CRT.

☞ Composées de 20 bits, repartis en 12 bits pour l'adresse et 8 bits pour spécifier l'opération, les instructions de l'IAS ne sont pas aussi larges que celles de l'EDVAC. Elles ont la forme suivante (fig. 4.28):

accumulateur <-- accumulateur + opérande(A), avec OP = «+»



OP: opération, A: adresse d'un opérande

Fig. 4.28 Répertoire d'instructions de l'IAS.

La machine présente 17 instructions différentes. Les données ayant un format de 40 bits, le mot mémoire présente ce même format. On peut ainsi loger 2 instructions par mot de 40 bits de la mémoire, soit lire la mémoire deux fois moins vite que le nombre d'instructions à exécuter. L'architecture de l'IAS, illustrée à la figure 4.29, est très proche de celle de Von Neumann (fig. 4.19), mais elle n'est toutefois pas organisée autour d'un seul bus [5]. C'est ici le registre DR (Data Register) qui reçoit les données ou instructions de la mémoire avant de les envoyer au compteur de programme (PC), au registre d'instructions, à l'unité arithmétique et logique (ALU), ou enfin au registre d'adresse de la mémoire (AR, Address Register).

Equipé de 2500 tubes à vide, l'IAS fonctionne pour la première fois en juin 1952. Même en étant en parallèle, avec une unité arithmétique comportant 40 fois plus de tubes qu'une machine série, l'IAS comporte moins de tubes qu'une machine série comme l'EDVAC. La raison est que la partie de contrôle d'une machine série est beaucoup plus compliquée que celle de l'IAS. Elle demande en effet de séquencer un à un les bits d'une donnée et d'une instruction, laquelle doit par ailleurs être lue dans sa totalité avant d'être exécutée.

L'Institute for Advanced Studies (IAS) existe toujours à Princeton, mais on n'y trouve aujourd'hui plus trace de Von Neumann ni de l'ordinateur IAS.

Les machines BINAC et UNIVAC

Après avoir quitté la Moore School, Eckert et Mauchly fondent leur propre compagnie. Ils débutent la construction d'autres machines selon le modèle de l'EDVAC, dont le BINAC (BINary Automatic Computer) pour la firme Northrop Aircraft, et l'UNIVAC (UNIVersal Automatic Computer) pour une agence gouvernementale.

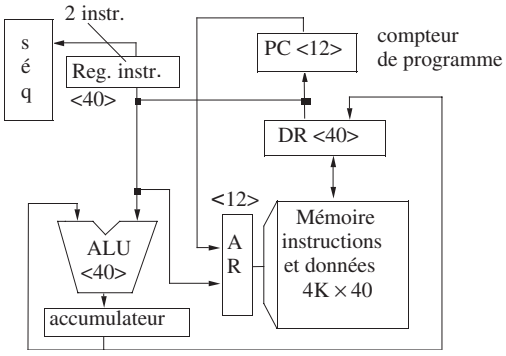


Fig. 4.29 Architecture de l'IAS.

Le BINAC présente deux unités de calcul en parallèle pour comparer les résultats et détecter des erreurs. Il utilise des lignes à retard à mercure comme mémoire. Terminé en septembre 1949, il est le premier ordinateur électronique à programme enregistré apparu aux Etats-Unis.

L'UNIVAC comporte quant à lui 45 instructions, dont deux logées dans un mot mémoire, comme dans l'IAS. L'instruction comporte une seule adresse, et quatre accumulateurs peuvent être utilisés. La mémoire comporte 1000 mots. La machine fonctionne à plus de 2 MHz (2 millions de pulsations par seconde) et comporte 5000 tubes à vide.

L'UNIVAC et le BINAC coûteront beaucoup plus cher que prévu, et la compagnie de Mauchly et Eckert tombe en faillite. Elle sera rachetée par Remington-Rand, qui va renégocier les contrats de l'UNIVAC. La machine est terminée en mars 1951, et une cinquantaine d'exemplaires sont vendus, permettant du même coup à Remington de devenir la principale compagnie d'ordinateurs.

NOM	TYPE	Année	Inventeur	Mémoire	Mémoire: type
EDVAC	tubes[1]	1845-1951	von Neumann	1024×44 bits	ligne à retard
EDSAC	tubes[1]	1947-1949	Wilkes	1024×17 bits	ligne à retard
BINAC	tubes[1]	1947-1949	[2]	512×31 bits	ligne à retard
UNIVAC	tubes[1]	1947-1951	[2]	1000×12 digits	ligne à retard
SEAC	tubes[1]	1948-1950	Alexander	512×44 bits	ligne à retard
Manchester	tubes[1]	1946-1948	Newman	32×32	CRT
IAS	tubes[1]	1947-1952	von Neumann	256×40 bits	CRT
SWAC	tubes[1]	1948-1952	Huskey	256×37 bits	CRT (UCLA)
WHIRL-WIND	tubes[1]	1947-1950	Forrester	256×16 bits	CRT (MIT)
	[1] électroniques		[2] Eckert, Mauchly		

Fig. 4.30 Classement des premiers ordinateurs électroniques.

Les ordinateurs en URSS

L'histoire du développement des premiers ordinateurs en URSS à la fin de la guerre demeure aujourd'hui encore méconnue, notamment en raison de l'absence totale de coopération entre l'Est et l'Ouest [11, 12].

Les réalisations russes les plus marquantes ont lieu dans le domaine du logiciel, et tout particulièrement dans le secteur des compilateurs. Sur le plan du matériel, les éléments électroniques étaient connus bien avant la guerre, puisqu'en 1918 un professeur russe du nom de M. A. Bonch-Bruevitch inventa la première bascule électronique réalisée avec des tubes à vide.

Le premier projet de développement d'un ordinateur en URSS est lancé en 1948 par les professeurs S.A. Lebedev and B.I. Rameev. Ils mettent au point la machine MESM en 1950 à l'Institut d'Electrotechnique de l'Académie des Sciences d'Ukraine, et elle sera exploitée dès l'année suivante. Cet ordinateur comporte 3800 tubes électroniques et utilise des instructions de 21 bits avec trois adresses, soit deux sources et une destination. Il peut adresser 31 registres de données, 63 instructions et 100 constantes uniquement en lecture. Ses performances en vitesse demeurent inconnues. D'autres machines suivent, comme le BESM, mis en service en 1952 à l'Institut des techniques de mécanique de précision et d'ordinateurs de l'Académie des Sciences d'URSS, capable d'exécuter 8000 instructions par seconde. Le M-2, apparaît la même année avec une performance de 2000 instructions par seconde, tout comme la machine «Strela» en 1953, et qui est la première à être produite en série.

La figure 4.31 représente quelques données relatives aux premiers ordinateurs soviétiques [11, 12]. Toutes ces machines ont des instructions comportant trois adresses, sauf l'Ural (une adresse) et le M-3 (deux adresses).

	BESM	M2	Strela	Ural	M3
Année	1952	1952	1953	1954	1956
Instr./seconde	8000	2000	2000	100	30
nb. bits	39	34	43	35	30
mémoire	tube élec.	tube élec.	tube élec.	magnétique	magnétique
nb. mots	1023	512	2048	1023	2048
nb. instructions	32	30	30	29	48
nb. tubes	4000	1670	8000	800	770
puissance	80 kW	29 kW	150 kW	7,5 kW	8 kW
surface m^2	170	22	150	60	20

Fig. 4.31 Ordinateurs en URSS.

Toutes ces machines sont assez similaires à celles construites en Occident, certaines présentant toutefois quelques innovations intéressantes.

L'ordinateur BESM, décrit succinctement dans l'annexe de la référence [13], présente notamment quelques caractéristiques surprenantes. Il utilise comme périphériques un tambour magnétique, quatre lecteurs de bandes magnétiques, un lecteur de cartes perforées en entrée et une imprimante en sortie. La mémoire comporte 1024 positions de 39 bits, ainsi que 384 positions en lecture seulement pour des constantes et pour mémoriser des sous-routines standard (librairie). Les instructions présentent trois adresses de 11 bits (deux sources, une destination), ainsi qu'un code opératoire de 6 bits (au total, 39 bits). Mais son caractère le plus insolite est la présence de deux compteurs de programme: l'un de 11 bits (dit «central») adresse les instructions et est incrémenté après chaque instruction opérative, l'autre (dit «local») est utilisé pour les sous-routines. Ce système permet de laisser l'adresse courante du programme principal dans le premier compteur, de charger l'adresse de la sous-routine dans le 2e compteur et d'utiliser celui-ci pour l'exécution de la sous-routine. Lors du retour au programme principal, on retrouve donc l'adresse courante dans le premier compteur. Ce mécanisme de sous-programme, introduit par Augusta Ada Byron (1815-1852) lors de la conception de la machine de Babbage (chap. 2), est concrétisé pour la première fois dans la réalisation du BESM.

☞ Les nombres sont eux aussi représentés avec 39 bits, en virgule flottante, avec 6 bits pour l'exposant (bit de signe compris), et 33 bits pour la mantisse (idem). L'unité arithmétique est donc en virgule flottante.
L'ordinateur comporte par ailleurs 39 instructions différentes, parmi lesquelles les opérations arithmétiques, les comparaisons et les décalages, les transferts de mémoire à mémoire, des instructions d'entrée/sortie, les instructions de branchement et d'appel des sous-routines. Une manière d'appeler une sous-routine est l'instruction CLCC, adr, laquelle permet de se brancher à l'adresse «adr» du début de la sous-routine chargée dans le 2e compteur de programme (fig. 4.32). On revient alors à l'adresse du programme principal stockée dans le premier compteur de programme grâce à l'instruction JCC. Une autre manière de procéder consiste à utiliser l'instruction CCCC, adr1, adr2, qui sauve l'adresse du compteur de programme dans la position mémoire d'adresse «adr1», et se branche à «adr2», adresse de la sous-routine. Pour le retour, plusieurs instructions du même type sont utilisées.

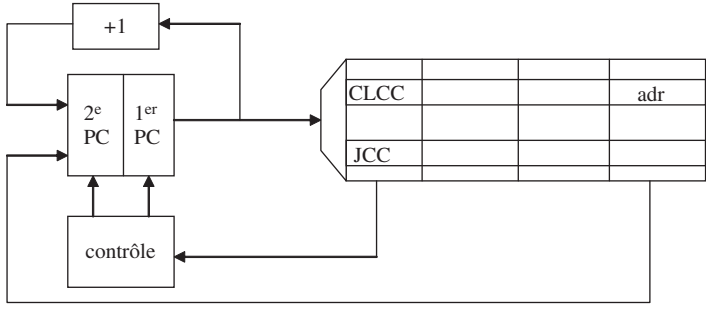


Fig. 4.32 Appel et retour de sous-routine dans le BESM.

L'URSS n'a jamais développé de filière originale en matière d'ordinateurs et s'est progressivement inspirée des machines mises au point par IBM.

Les premiers ordinateurs transistorisés (Razdan, Dnepr, Minsk-2) apparaissent en 1960 en URSS, soit 2 ans après le 7090 transistorisé conçu par IBM. En 1966, la plus performante des machines russes, le BESM-6, atteint la performance de 1 million d'instructions par seconde [17]. Six ans plus tard, le projet «EC» est lancé, inspiré directement des ordinateurs IBM 360. Le BESM-6 qui sera cependant utilisé jusqu'en 1980, date à laquelle sont développés d'autres ordinateurs rapides tels que El'brus, SVS-1, Mars et ES-1766, pour la plupart inspirés de machines occidentales [17]. Mais les progrès technologiques occidentaux sont si rapides, que le retard de l'URSS en matière de développement d'ordinateurs ne cesse de s'accroître.

L'URSS a cependant brillé dans le développement de logiciels, en réussissant à mettre au point dès 1953 des compilateurs capables de traduire un langage de haut niveau proche des algorithmes en un langage compréhensible par la machine.

Le professeur Lyapunov fut le principal artisan de cette réussite [12]. Le premier compilateur (PP-1, pour Programming Program), conçu pour l'ordinateur Strela, est terminé en 1954, suivi du PP BSEM en 1956. D'autres compilateurs sont développés pour les ordinateurs Strela-3 et Strela-4, les PPS et PP-Strela. La méthode introduite par Lyapunov consiste à modéliser un compilateur par un opérateur complexe qui agit sur les positions mémoire de l'ordinateur en représentant une séquence d'opérations élémentaires [13].

En Chine la première génération d'ordinateurs est copiée de ceux d'URSS, dans le cadre d'un programme d'assistance URSS-Chine [15].

Le Ural-2, mis au point dans ce cadre, fonctionne dès 1959, soit avant même l'original dont il est inspiré! En 1963, le BSEM-2 devient en Chine le DJS-2. Le pays construit ensuite ses propres ordinateurs, dont certains directement inspirés des séries 360 de IBM.

La microprogrammation

Le microprogramme, conçu par Babbage (chap. 2), est un programme à l'intérieur de la machine exécutant une seule instruction en plusieurs étapes. L'adoption de ce principe dans les unités de commande des ordinateurs va prendre cependant quelque temps.

Néanmoins les unités de commande des premiers ordinateurs ne sont pas microprogrammées; elles sont constituées d'éléments logiques tels que des tubes, puis des transistors, selon une logique dite «câblée» n'utilisant pas de mémoires de microprogramme. En 1951, le professeur M. V. Wilkes de l'Université de Manchester introduit le concept de microprogrammation pour les unités de commande [9]. Chaque instruc-

tion d'une machine devant être séquencée en un certain nombre d'opérations élémentaires, Wilkes propose d'appeler celles-ci des micro-opérations, l'ensemble des micro-opérations nécessaires à l'exécution d'une instruction constituant un microprogramme.

☞ La figure 4.33 représente la structure d'une unité de commande microprogrammée. Le registre comporte l'adresse d'une micro-instruction. Après décodage, cette adresse sélectionne une ligne horizontale, et par conséquent une configuration de «0» et de «1», dans le réseau A qui va commander l'unité de traitement. Le réseau B produit de même l'adresse de la prochaine micro-instruction qui sera chargée dans le registre. La prise en compte d'une alternative s'effectue en testant une condition provenant de l'unité de traitement, et en sélectionnant l'une ou l'autre des adresses (illustré par les deux dernières lignes du réseau B).

Conceptuellement, cette structure représente une mémoire dans laquelle sont mémorisées des micro-instructions. Le problème majeur qui se posait aux chercheurs en 1950 était la réalisation même de ces mémoires, ainsi que la vitesse de lecture. La microprogrammation fut utilisée pour la première fois dans l'EDSAC2 à Cambridge, mais il fallut attendre 1961 pour que le premier ordinateur commercial microprogrammé, le TRW 130, soit mis au point.

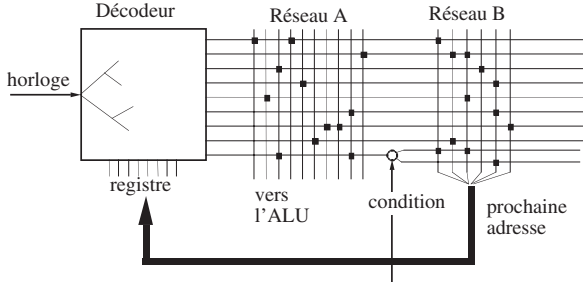


Fig. 4.33 Microprogrammation proposée par Wilkes.

La commercialisation d'ordinateurs

La construction d'ordinateurs commerciaux débute avec l'UNIVAC et se poursuivra avec des machines de plus en plus puissantes.

Parmi les machines les plus fameuses, citons la série 700 d'IBM. La première fut le 701, en 1952, avec 33 instructions, suivi du 704 deux ans plus tard, doté de 75 instructions différentes. Le transistor fut inventé fin 1947, et apparut dès 1958 dans les ordinateurs UNIVAC et IBM, tels que l'IBM 7090 (un 709 transistorisé). La vitesse était augmentée d'un facteur 10 par rapport aux tubes à vide. Après la série 1400, IBM annonce en 1964 celle des IBM 360, machines qui feront la réputation de la compagnie. D'autres sociétés comme Burrough, Control Data, General Electric, Honeywell, NCR, RCA construisent parallèlement de très gros ordinateurs, mais ne parviennent

pas à déloger IBM de sa position de leader sur le marché [8]. La série 360 révolutionne l'ordinateur dans la mesure où IBM propose des machines de puissances et de prix différents entièrement compatibles entre elles, c'est-à-dire capables d'exécuter un même programme: les ordinateurs les moins puissants exécutent simplement le programme plus lentement. C'est un immense progrès pour l'utilisateur, qui n'a plus alors à ré-écrire tous les programmes contenus dans sa vieille machine avant de pouvoir la remplacer par un modèle plus récent.

En 1971, au sommet de sa puissance, IBM lance la série 370, mais Digital Equipment Corp. introduit peu de temps après les mini-ordinateurs, tels que le PDP 11, qui marquent la fin du monopole de IBM. Cette même année est inventé le premier microprocesseur, le 4004 d'Intel. L'arrivée des ordinateurs personnels (PC), Apple II, PET ou IMSAI au milieu des années 1970 est quant à elle synonyme de déclin pour Digital Equipment Corp. et d'autres entreprises.

Les coûts et performances des ordinateurs

Si l'on compare l'évolution des coûts et performances des ordinateurs et ceux des voitures, on pourrait acheter aujourd'hui une Rolls pour moins d'un franc! La table suivante illustre l'évolution des performance rapportées au coût:

1951	Tube à vide	1
1965	Transistor	35
1975	Circuit intégré	900
1990	VLSI	400 000

L'inventeur de l'ordinateur

Qui est donc l'inventeur de l'ordinateur? Un procès a opposé Von Neumann à Mauchly et Eckert à ce sujet. En 1971, la justice américaine accorda finalement le brevet de l'ordinateur à Atanasoff, et non aux concepteurs de l'ENIAC [2]. C'est toutefois Zuse qui est aujourd'hui considéré comme son véritable inventeur.

Aucun de ces brillants inventeurs et précurseurs n'aura cependant prédit le succès que l'ordinateur a rencontré plus tard. Il est en effet longtemps demeuré une machine dont l'utilité se limitait à la résolution de longs calculs, et ce n'est que très récemment que les ordinateurs personnels, considérés comme gadgets il y a 10 ans encore, ont massivement envahi notre vie quotidienne.

Les concepts

Il est remarquable de constater que beaucoup des concepts utilisés aujourd'hui encore dans les microprocesseurs modernes ont été inventés à l'époque des premiers ordinateurs électroniques [10]:

- *La séquence de commandes*
  Hérité de la mécanique (notamment de la boîte à musique) et réalisé en électronique, ce concept est toujours utilisé aujourd'hui, sous le nom de séquenceur. La séquence de commandes est utilisée dans les unités de commande de certains circuits intégrés.
- *Le programme enregistré*
  Attribué à Von Neumann, ce concept définit l'ordinateur.
- *L'architecture Von Neumann*
  Ce concept signifie que instructions et données sont logées dans la même mémoire. C'est ainsi que sont réalisés aujourd'hui l'immense majorité des microprocesseurs.
- *Les instructions composées d'un seul mot de même longueur, puis CISC série 360*
  Tous les premiers ordinateurs possédaient cette caractéristique. L'utilisation d'une seule mémoire pour données et instructions mena à l'apparition d'instructions composées de plusieurs mots mémoire et d'architectures «CISC», telles que celles de la série 360 de IBM. On redécouvrit le concept d'instructions composées d'un mot de même longueur dans les années 1980 lors de l'arrivée des machines RISC. Ce type d'instructions présentait en effet de nombreux avantage en matière de simplicité et de vitesse d'exécution.
- *Les machines asynchrones*
  L'IAS de Von Neumann fonctionnait sur ce concept. Mais ce n'est que très récemment que l'on vit apparaître de tels microprocesseurs, dont l'avantage est de fonctionner à leur propre rythme et de se passer d'horloge de base.
- *Un accumulateur, plusieurs accumulateurs, le banc de registres*
  Toutes les machines actuelles ont adopté l'une de ces trois solutions, déjà en vigueur dans les premiers ordinateurs.
- *L'importance de la technologie mémoire, la mémoire série*
  Les mémoires constituent un perpétuel facteur limitant des performances des microprocesseurs. Leur technologie a toujours influencé les architectures: une mémoire série demande par exemple une arithmétique série.
- *Les mémoires parallèles, la hiérarchie de mémoires, la mémoire cache du Colossus II*
  Certaines idées proposées pour les premiers ordinateurs sont appliquées partout aujourd'hui. Ainsi les caches, petites mémoires rapides entre le processeur et la mémoire principale.
- *La détection de fautes par redondance*
  Expérimenté pour la première fois par Eckert et Mauchly lors de la réalisation du BINAC, ce concept est aujourd'hui utilisé pour les mémoires et les microprocesseurs complexes, essentiellement dans les applications sensibles (espace, avionique).

– *Les machines compatibles IBM 360*
  Un tel concept est encore en vigueur aujourd'hui, le meilleur exemple étant celui des microprocesseurs Intel 86, 186, 286, 386, 486, Pentium, PentiumPro et Itanium capables d'exécuter le même code x86.
– *La microprogrammation, IBM 360*
  Ce concept a été utilisé jusqu'aux microprocesseurs 68 000 à 68 060 équipant les vieux MacIntosh.
– *Le pipeline*
  L'IBM 360/91 fut le premier ordinateur construit sur ce concept. Tous les microprocesseurs actuels sont réalisés avec un pipeline, que l'on pourrait comparer à une chaîne de montage où se succèdent les instructions. Chacune d'elle commence à être exécutée avant même que la précédente ne soit terminée.

Tous ces concepts seront repris dans le chapitre 6.

Références

[1] M. R. Williams, *History of Computing Technology*, IEEE Computer Society Press, Los Alamitos, CA, second edition, 1997.

[2] D. Rutland, *Why Computers are Computers*, Wren Publishing, 1995.

[3] Philippe Breton, *Une histoire de l'informatique*, Coll. Points S65, Seuil, 1990.

[4] Georges Ifrah, *Histoire universelle des chiffres*, Bouquins, Robert Laffont, 1994.

[5] John P. Hayes, *Computer Architecture and Organization*, McGraw-Hill Book Company, 1978.

[6] Mario de Blasi, *Computer Architecture*, Addison-Wesley, 1990, p. 26.

[7] «Qui a inventé l'ordinateur?» *Les Cahiers de Sciences & Vie*, Hors Série n° 36, Décembre 1996.

[8] B. L. Peuto, *Mainframe History Provides Lessons*, Microprocessor Report, March 31, 1997.

[9] M. V. Wilkes, «The Best Way to Design an Automatic Calculating Machine», Paper presented at the Manchester University Computer Inaugural Conference, July 1951.

[10] C. Piguet, «Are Early Computer Architectures a Source of Ideas for Low-Power?», Volta'99, Como, Italy, March 4-5, 1999.

[11] Nicolas Starodoubtsev, Saint-Pétersbourg, communication privée, 1999.

[12] A. P. Ershov *et al.* «Computer Development in the USSR», XXXI Diebold Conference, Rome, June 11-13, 1974; International Research Conference on the History of Computing, Los Alamos, June 10-15, 1976.

[13] A. P. Ershov, *Programming Programme for the BESM Computer*, Pergamon Press, 1959.

[14] Raul Rojas, «Konrad Zuse's Legacy: The Architecture of the Z1 and Z3», IEEE Annals of the History of Computing, vol. 19, n° 2, 1997, pp. 5-16.

[15] H. L. Gardner, «Computing in China, 1978», IEEE Computer, March 1979, pp. 81-96.

[16] Simon Singh, *Histoire des codes secrets*, JC Lattès, 1999.

[17] P. Wolcott, S. E. Goodman, «High-Speed Computers of the Soviet Union», IEEE Computer, vol. 21, n° 9, September 1988, pp. 32-41.

[18] D. M. Yates, «Turing's Legacy, A history of computing at the National Physical Laboratory 1945-1995», Science Museum, Exhibition Road, London.

CHAPITRE 5

HISTOIRE DE LA PROGRAMMATION

«La programmation est la branche la plus difficile des mathématiques appliquées.»
Edsger W. Dijkstra

Programmation

Les instructions de l'ordinateur sont constituées de 0 et de 1. Et comme avec les goupilles du cylindre de la boîte à musique, ces 0 et ces 1 se trouvent dans la mémoire de l'ordinateur et permettent de programmer une machine.

Dans la boîte à musique, la programmation d'une mélodie consiste à dresser la liste des goupilles à insérer sur le cylindre. Elle est relativement simple. Comme chaque goupille actionne un son bien défini, il y a correspondance directe entre goupille et son. L'arrangement des goupilles reflète celui des notes et le cylindre est ainsi l'image de la partition. La programmation consiste donc en la transcription simple et directe des notes de la partition musicale en une liste de goupilles à disposer sur le cylindre.

Dans l'ordinateur, la correspondance entre un calcul à effectuer et les 0 et 1 stockés en mémoire n'est pas si élémentaire. La distance est même grande entre ces codes et le programme d'une tâche écrit dans un langage de programmation comme C ou Java. Et elle est encore bien plus grande si l'on mesure tout le chemin qui mène de l'idée que se fait l'utilisateur de la tâche à entreprendre jusqu'à cette représentation en codes élémentaires stockés dans la mémoire de l'ordinateur. La programmation d'un ordinateur consiste justement à parcourir ce chemin.

Impensable avant l'existence même des ordinateurs, la programmation naît du besoin et se développe dans l'urgence d'exploiter les machines nouvellement créées en cherchant à leur confier des tâches utiles et économiquement intéressantes. Aujourd'hui, la programmation est une activité économique très importante et une discipline scientifique dont les défis à relever demeurent nombreux.

Programmer, c'est réaliser une idée sur un ordinateur.

La figure 5.1 illustre le chemin à parcourir pour programmer un ordinateur. Il conduit d'une idée à réaliser à l'exécution du code machine par l'ordinateur. L'idée est immatérielle et réside dans l'esprit du concepteur alors que le code prend la forme d'une suite de 0 et de 1 déposés dans la mémoire de l'ordinateur. Trois éléments jouent un rôle essentiel sur le chemin qui va de l'idée à sa réalisation par l'ordinateur: l'algorithme, le langage de programmation et le système d'exploitation.

Pour programmer il faut d'abord un algorithme.

L'algorithme est le premier élément, et constitue l'étape initiale vers cette réalisation. Face à une idée, qui est parfois vague, généralement ouverte sur plusieurs possibilités et sujette à des imprécisions, il est nécessaire de prévoir la manière de la réaliser. Il s'agit d'établir la liste précise des opérations à effectuer pour réaliser cette idée. L'algorithme constitue cette liste. Quand arrivent les premiers ordinateurs, les programmeurs ne sont pas totalement pris au dépourvu, car l'algorithmique existe depuis longtemps. On sait même qu'elle existait déjà durant l'Antiquité.

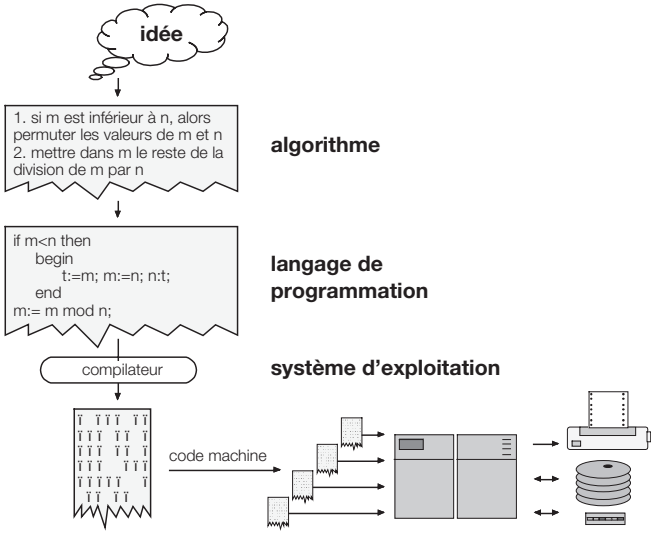


Fig. 5.1 La programmation conduit de l'idée à sa réalisation sur l'ordinateur.

Pour programmer il faut ensuite un langage de programmation.

Le langage de programmation représente la deuxième étape. Il a pour objet la forme utilisée pour communiquer cet algorithme à l'ordinateur. De nos jours, on fera usage selon les besoins d'un langage comme C++ ou Java, ou encore de l'un des 2350 autres langages de programmation créés avant 1997 [20]. Bien que beaucoup d'entre eux soient peu importants ou inutilisés, de nombreux langages sont dignes d'intérêt. L'histoire des langages de programmation va de pair avec celle des ordinateurs ([5][10][11][12][16][19][20]). On s'intéressera ici à l'histoire de quelques langages majeurs, que ce soit parce qu'ils ont réussi à s'imposer ou parce qu'ils ont eu une influence remarquable sur d'autres langages.

Pour programmer, il faut finalement un système d'exploitation.

Le système d'exploitation que constitue l'ensemble des moyens et des outils logiciels mis en place pour faciliter l'exécution d'un programme est un autre élément important. Il faisait complètement défaut lors de l'apparition des ordinateurs et près de 10 ans furent nécessaires pour que les premiers systèmes d'exploitation voient le jour. Le développement qui a suivi n'en fut que plus rapide.

Les algorithmes

L'algorithme est à la programmation ce que la recette est à la cuisine: une succession d'étapes à parcourir pour obtenir un résultat.

Confronté à une tâche comportant des degrés de liberté, l'homme exerce généralement son bon sens pour se guider dans les différentes étapes du travail. Il réussit ainsi à réaliser avec succès divers travaux, mêmes si ceux-ci sont vagues. Pour être réalisable par un automate, une tâche doit décrire précisément les étapes de sa réalisation et ne peut supporter des degrés de liberté. La liste des étapes destinées à résoudre un calcul ou autre problème est appelée algorithme. Plus formellement, un algorithme est une séquence finie d'instructions, chacune ayant une signification claire et pouvant être exécutée avec des ressources limitées.

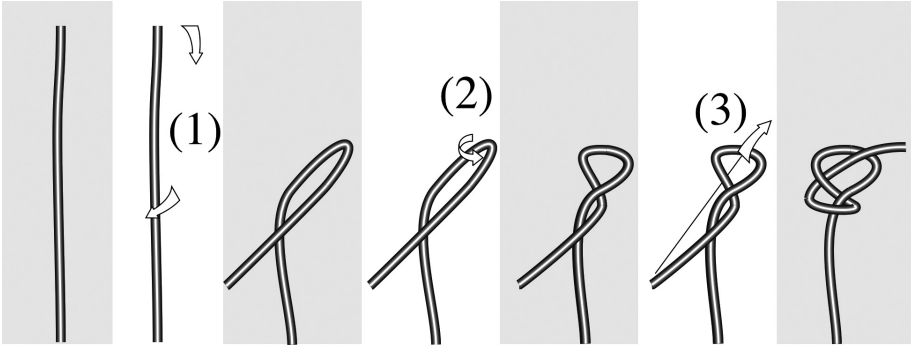


Fig. 5.2 Un algorithme commun.

Un algorithme n'est pas forcément destiné à un ordinateur. Il peut être employé pour formaliser des tâches plus communes, telle celle illustrée par la figure 5.2. Il requiert néanmoins chaque fois l'utilisation d'instructions parfaitement claires.

Les premiers algorithmes

Durant l'Antiquité grecque, les mathématiques étaient essentiellement perçues en termes d'algorithmes, c'est-à-dire d'opérations à effectuer pour obtenir certains résul-

tats. Pour décrire ces algorithmes, il était alors fait usage du langage naturel mais sous une forme stylisée ressemblant à un langage de programmation.

☞ Voici un exemple simplifié tiré de Louden, et décrivant une manière systématique de calculer la profondeur d'une citerne à partir de ses dimensions [8]:

ceci est l'énoncé:	ceci est la procédure:
– une citerne	
– la longueur vaut 10	(1) diviser 120 par 10 pour obtenir 12
– la largeur vaut 4	(2) diviser 12 par 4 pour obtenir la profondeur, 3
– le volume excavé vaut 120	
– que vaut la profondeur?	

D'autres algorithmes ont été inventés, notamment par les mathématiciens Euclide et Eratosthène, pour résoudre des problèmes plus abstraits.

Euclide

Le mathématicien grec Euclide (IV^{e} siècle av. J.-C.) est un précurseur dans la formulation d'algorithmes. Dans son œuvre *Les Eléments* qui contient les fondements de la géométrie, on trouve notamment un algorithme destiné au calcul du plus grand commun diviseur de deux nombres.

☞ Cet algorithme s'énonce comme suit:
plus grand commun diviseur de m et de n
(1) si m est inférieur à n, alors permuter les valeurs de m et n
(2) mettre dans m le reste de la division de m par n
(3) si m est différent de zéro, alors retour au pas 1 avec les nouvelles valeurs de m et n
(4) n contient le résultat

La table qui suit illustre les différentes étapes de l'algorithme pour un choix initial de m = 18 et n = 24. On voit que les étapes 1, 2 et 3 sont parcourues deux fois et que l'algorithme finit avec l'étape 4, dont le résultat escompté est n = 6.

étape	m	n
0	18	24
1	24	18
2	6	18
3	6	18
1	18	6
2	0	6
3	0	6
4	0	6

Eratosthène

Eratosthène est un mathématicien, astronome et philosophe grec de l'école d'Alexandre. Il gagne la postérité en réussissant à mesurer le méridien terrestre. Il est

aussi l'auteur d'un algorithme permettant de déterminer les nombres premiers. Est nombre premier tout entier plus grand que 1 qui n'est divisible que par 1 et par lui-même.

☞ Le crible d'Eratosthène trouve systématiquement les nombres premiers dans la liste des nombres qui vont 2 à n (fig. 5.3):

(1) k:=2
(2) pour chaque nombre, m, compris entre k+1 et n faire
si m est un multiple de k, alors biffer m de la liste des nombres
(3) mettre dans k le premier nombre non biffé qui lui est supérieur
(4) si k est inférieur à n alors continuer en 2
(5) tout nombre qui n'a pas été biffé est un nombre premier

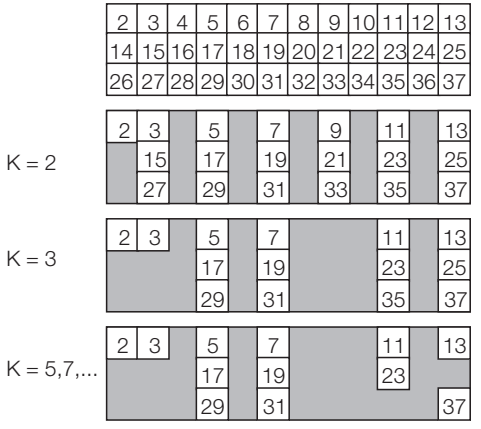


Fig. 5.3 Les étapes du crible d'Eratosthène pour n = 37.

Al-Khowarizmi

Le mathématicien arabe Al-Khowarizmi enseigne à Bagdad au début du IX^{e} siècle de notre ère. Il contribue à la dissémination du système décimal en reportant, entre autres, la découverte par les Indiens du système décimal avec un zéro dans son livre intitulé *Al-jabr wa'l muqabala*, titre dont provient le terme «algèbre». Le propre nom de ce mathématicien est quant à lui à l'origine du terme «algorithme».

Les logarithmes

La technique du calcul à base de logarithmes est décrite par Napier dans son ouvrage *Mirifici logarithmorum canonis descriptio*, publié en 1614. Les logarithmes permettent notamment de réduire les multiplications et divisions aux opérations plus simples que sont l'addition et la soustraction.

Le principe repose sur les égalités suivantes, montrant que le produit $p \times q$ est réalisable par la somme de log(p)+log(q) à condition de maîtriser les transformations logarithmiques.

$$p \times q = b^{\log(p)} \times b^{\log(q)} = b^{\log(p)+\log(q)}$$

Le passage d'un nombre à son logarithme de même que l'opération inverse nécessitent de consulter une table. Napier établit des tables pour la base particulière b= 2,718... sans tenir compte que ce choix n'offre pas une généralisation dans l'utilisation des tables. C'est un Anglais contemporain de Napier, Henry Brigg, qui découvre les avantages de la base b=10 et qui en dressera les tables. La seule table pour les nombres compris entre 1 et 10 est ainsi suffisante pour faire la conversion, comme l'illustre l'exemple ci-dessous:

$$\begin{aligned} \log_{10} 1234 = \log_{10} (1{,}234 \times 1000) &= \log_{10} 1{,}234 + \log_{10} 1000 \\ &\approx \log_{10} 1{,}23 + 3 \\ &\approx 3{,}0899 \end{aligned}$$

Hilbert

On sait donc depuis fort longtemps que l'algorithmique est un bon moyen de résoudre les problèmes. Mais ce n'est qu'au début du XX^{e} siècle que des questions plus fondamentales apparaissent.

Le grand mathématicien David Hilbert suggère que tous les problèmes mathématiques devraient être encodés selon un formalisme approprié et qu'un algorithme devrait être trouvé pour en établir la solution. Il expose notamment l'*Entscheidungsproblem* au Congrès international des mathématiciens en 1928. C'est au cours de cette réunion qu'il soulève l'éventualité de l'existence d'une méthode ou d'un processus mécanique déterminé selon lesquels toutes les questions mathématiques pourraient être résolues.

Gödel et la décidabilité

La question de savoir si un problème peut être résolu, appelée encore décidabilité, est fondamentale. Elle se pose en fait aussi pour des problèmes simples. On peut par exemple se demander si l'algorithme suivant termine pour tous les entiers x:

(1) tant que $x \neq 1$ faire
  (1.1) si x est pair alors x: = x/2
  (1.2) si x est impair alors x: = 3x + 1
(2) stop

On constate que l'algorithme produit une suite de x qui vont croissant quand x est impair et décroissant quand x est pair. L'algorithme se termine pour tous les nombres x essayés... mais est-ce le cas pour tous les nombres?

La décidabilité n'est en fait pas toujours possible, comme en apporte la preuve Kurt Gödel en 1931. Il montre que même pour une classe de problèmes relativement simples, le calcul des prédicats du premier ordre, la décidabilité ne peut être toujours établie. Cette contribution est connue sous le nom de théorème d'incomplétude de Gödel.

Alan Turing et la calculabilité

De fait, le théorème de Gödel sur la décidabilité n'est que la moitié de la réponse à la question d'Hilbert. L'autre moitié concerne la calculabilité. Il s'agit de déterminer, face à un problème décidable, quelle sorte de machine est nécessaire à sa résolution, ou quel type de machine est en mesure de résoudre un problème donné.

La réponse à cette question est fournie par Alan Turing. En 1936, ce mathématicien britannique publie un article intitulé «*On Computable Numbers with an Application to the Entscheidungsproblem*». Ce travail est remarquable, car le concept de «Machine de Turing» qui y est exposé devient le fondement de la théorie de l'informatique. Turing montre que ce qu'on appelle calculable peut être réalisé par une machine extrêmement simple comprenant essentiellement une bande, un mécanisme de lecture écriture et un programme: la machine de Turing. Celle-ci est un dispositif simple et imaginaire pouvant réaliser n'importe quel algorithme solvable. En d'autres termes, tout problème solvable sur n'importe quel ordinateur est aussi solvable sur une machine de Turing. Cette conclusion, connue sous le nom de thèse de Turing et Church, indique que les ordinateurs sont équivalents entre eux dans leur potentiel à résoudre un problème [4]. Ou encore, qu'un mécanisme aussi simple qu'une machine de Turing peut simuler tous les mécanismes.

Turing n'a pas construit mais seulement imaginé sa machine. Cette architecture ne fut cependant jamais utilisée.

Si l'idée qu'une machine simple peut simuler toute machine actuelle ou future est philosophiquement importante, elle n'apporte pas la réponse à tout problème de calcul. Ainsi, l'efficacité de calcul varie très fortement d'un ordinateur à l'autre et la machine de Turing ne présente pas d'avantage sur ce point. L'efficacité de calcul peut faire toute la différence entre deux ordinateurs, comme le savent bien leurs constructeurs.

La machine de Turing

La machine de Turing consiste en une bande de longueur illimitée comprenant des cellules, ainsi qu'un contrôleur comportant un nombre limité d'états et capable de lire et écrire les symboles de la bande (fig. 5.4). Le contrôleur agit en fonction de l'état courant de la machine et du symbole lu en activant une des règles dont l'ensemble constitue le programme. Il peut surécrire le symbole courant et passer à un autre état. Il peut en outre se déplacer à gauche ou à droite sur la bande.

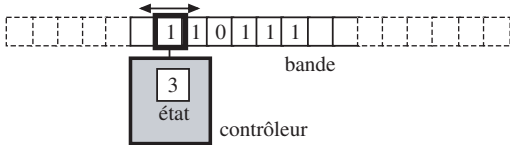


Fig. 5.4 Machine de Turing.

☞ La figure 5.5 montre deux exemples qui illustrent le fonctionnement de cette machine. A gauche (a), elle est programmée pour additionner deux nombres. Dans la situation initiale décrite en haut, les nombres deux (11) et trois (111) sont inscrits sur la bande, séparés par un 0, et la machine est positionnée tout à gauche. Les nombres sont en représentation unaire, c'est-à-dire que la valeur se lit en comptant le nombre de 1. A partir de cette situation initiale, la machine se comporte en appliquant successivement la règle valable pour l'état courant et le symbole lu. La figure illustre les déplacements et modifications de symboles successifs qui ont lieu et qui permettent d'arriver finalement au résultat escompté, disponible sur la bande quand la machine s'arrête, soit cinq (11111) dans l'exemple.

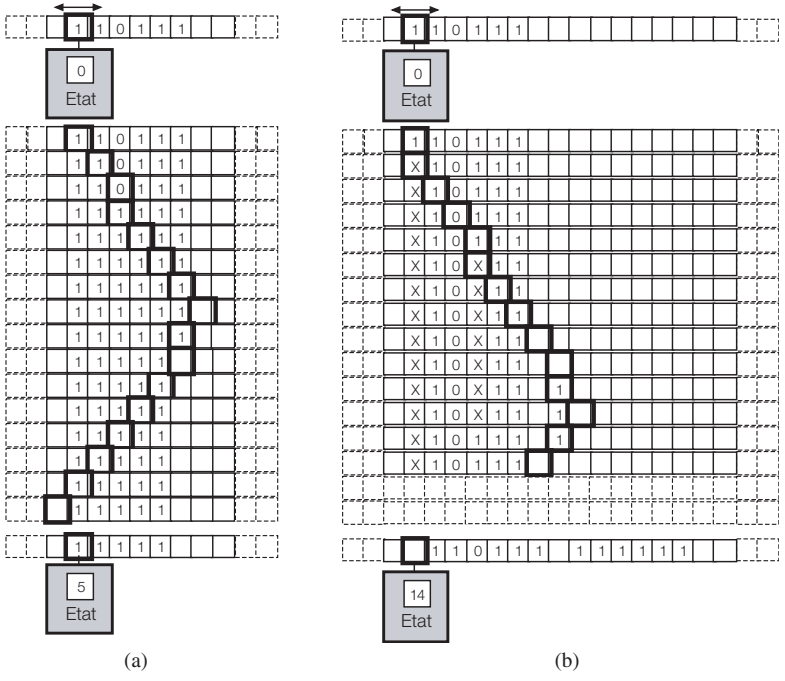


Fig. 5.5 Machines de Turing:
(a) somme de deux nombres unaires; (b) produit de deux nombres unaires.

L'exemple 5.5 (b) illustre le produit de deux nombres. Avec les nombres deux et trois sur la bande au moment du départ, la situation initiale représentée est la même que dans le cas précédent. Le programme comportant maintenant de nouvelles règles, la machine se comporte différemment, et produit successivement les nouveaux déplacements et modifications de symboles illustrés. La séquence des pas est alors plus longue. Toutes les étapes du calcul ne peuvent être représentées, la figure se limitant aux premières d'entre elles. La situation illustrée au bas de la figure est celle de la machine arrêtée. Elle contient, tout à droite sur la bande, le résultat six (111111) recherché.

Les concepts

Les concepts fondamentaux de l'algorithmique remontent à l'Antiquité et des mathématiciens fameux comme Euclide ou Eratosthène ont formulé des solutions à des problèmes sous la forme d'algorithmes. Les premiers programmeurs n'ont eu qu'à puiser dans ces méthodes de résolution pour écrire leurs premières applications.

Les concepts de décidabilité et de la calculabilité sont nés durant la première moitié du XX^{e} siècle. Les contributions de Gödel sur la décidabilité et de Turing sur la calculabilité jettent les fondements de la théorie de l'informatique. Tant les fondements théoriques que les algorithmes existent donc déjà lorsqu'apparaissent les premiers ordinateurs.

Les langages de programmation

Une fois l'ordinateur inventé et la tâche à résoudre exprimée par un algorithme, demeure le problème de la transmission de l'information à la machine…

On utilise de nos jours un des nombreux langages de programmation à disposition pour écrire et communiquer l'algorithme à l'ordinateur. Mais il n'en a pas été toujours ainsi, les procédés de communication ayant pris des formes diverses au cours du temps. Ils apparaissent tout d'abord sous la forme de simple transmission de bits et de code machine, avant d'évoluer vers la forme symbolique de langages assembleurs et d'aboutir à de véritables langages de programmation [9]. Ces derniers connaîtront ensuite plusieurs phases d'évolution [5][12].

Les étapes vers les langages de programmation

De ses premiers balbutiements à nos jours, la communication avec l'ordinateur a connu de nombreuses mutations.

Avant les années 1940, les ordinateurs étaient câblés. Toutes les liaisons de l'ordinateur étaient alors assurées par des connections établies soit au moyen de fils, soit au moyen d'interrupteurs. Le programmeur communiquait avec l'ordinateur en actionnant les commutateurs ou en déplaçant les fiches des fils, et travaillait au niveau du bit.

Ce travail était fastidieux, une action distincte étant nécessaire à la communication de chaque bit d'information à l'ordinateur. Cette forme de communication primitive avec l'ordinateur était alors loin de constituer un langage.

Une évolution majeure dans l'architecture des ordinateurs apparaît lorsque John von Neumann propose qu'une série de codes mémorisés dicte la séquence des actions à effectuer par l'unité de traitement. Les instructions sont stockées en mémoire sous la forme de séries de bits. La communication se faisait alors au travers de cette liste de motifs binaires, tout d'abord introduite en agissant successivement sur une simple lignée d'interrupteurs, puis plus tard au moyen d'un lecteur de bandes perforées. Cette forme de communication avec l'ordinateur via une notation binaire incompréhensible pour l'homme ne peut pas non plus être qualifiée de langage.

Peu après l'arrivée des premiers ordinateurs, les programmeurs réalisent l'intérêt d'une représentation plus compréhensible et de l'attribution d'un nom symbolique à chaque code d'instruction ainsi qu'à chaque place mémoire. La communication avec la machine au travers d'une suite d'instructions exprimée par des symboles ressemble alors à ceci:

```
LDA     #2
ADDA    Y
STA     X
```

Cette forme de communication prend le nom de «langage assembleur». Mais celui-ci souffre encore de nombreux inconvénients: il est dépendant de la machine, de bas niveau et demeure difficile à écrire et à comprendre.

Les programmeurs songent alors à la mise au point d'une formulation comportant un niveau d'abstraction plus élevé et permettant d'écrire des instructions sous une forme concise, compréhensible, universelle et exécutable sur toutes les machines. Ce souci concerne tout particulièrement certaines constructions standard telles que les assignations, les boucles et les sélections. Ainsi naît l'idée d'un langage, exprimant par exemple l'équivalent des trois instructions assembleur ci-dessus (c'est-à-dire mettre dans x la somme 2 + y) de manière simple par:

```
X: = Y + 2;
```

Les véritables langages de programmation sont nés. Ils se définissent comme une notation symbolique décrivant un calcul sous une forme lisible à la fois par l'homme et par la machine.

Programmer le premier ordinateur électronique

Voici le programme, exécuté le 21 juin 1948 par «The baby», premier ordinateur électronique à programme enregistré ayant fonctionné [21].

La mémoire de cette machine ne comprend que 32 mots à 32 bits. L'unité arithmétique opère sur des nombres entiers signés représentés en binaire par le complé-

ment à 2. «The baby» n'est capable que d'effectuer des soustractions. Il y a deux registres, le compteur d'instructions et un accumulateur. L'ensemble tourne à une vitesse d'exécution d'une instruction toutes les 1,2 ms.

Le premier programme du premier ordinateur électronique

Un des problèmes que le concepteur de «The baby», Tom Kilburn, choisit de programmer pour tester le fonctionnement de sa machine est celui du plus grand diviseur (pgd), une question arithmétique simple faisant appel à une recherche. Il consiste à chercher le plus grand nombre qui divise un nombre entier donné a. Le jour du premier test, le nombre a fut choisi petit pour arriver rapidement à la fin du programme. Au bout de quelques jours déjà, le programme fut testé pour a = 218. La réponse correcte fut trouvée en 52 minutes après que la machine eut effectué environ 2,1 millions d'instructions (voir aussi l'annexe).

```
function pgd(a: integer): integer;
var x, b: integer;
begin
b:=a;
 repeat
  x:=a;
  b:=b-1;
  while x>0 do x:=x-b;
 until x=0;
 pgd:=b;
end;
```

Fig. 5.6 Le programme pgd en Pascal.

La recherche du pgd consiste à tester la divisibilité de tous les nombres b plus petits que a. Réaliser ce test de divisibilité au moyen de la seule soustraction disponible consiste à soustraire b de a aussi longtemps que la différence est positive. Si la différence finale est nulle, b divise a. En Pascal, le programme correspondant s'écrirait par exemple selon la figure 5.6. Le programme de Kilburn avait quant à lui la forme manuscrite reproduite à la figure 5.7. Pour l'exécuter sur «The baby», il fallait introduire les codes binaires des deux dernières colonnes dans la mémoire de l'ordinateur.

Programmer en code machine

L'exemple de Kilburn montre qu'au début des années 1950, les programmeurs devaient d'abord transcrire leur algorithme au moyen des instructions disponibles sur la machine, avant d'établir la liste des codes binaires correspondants. Il devait être très

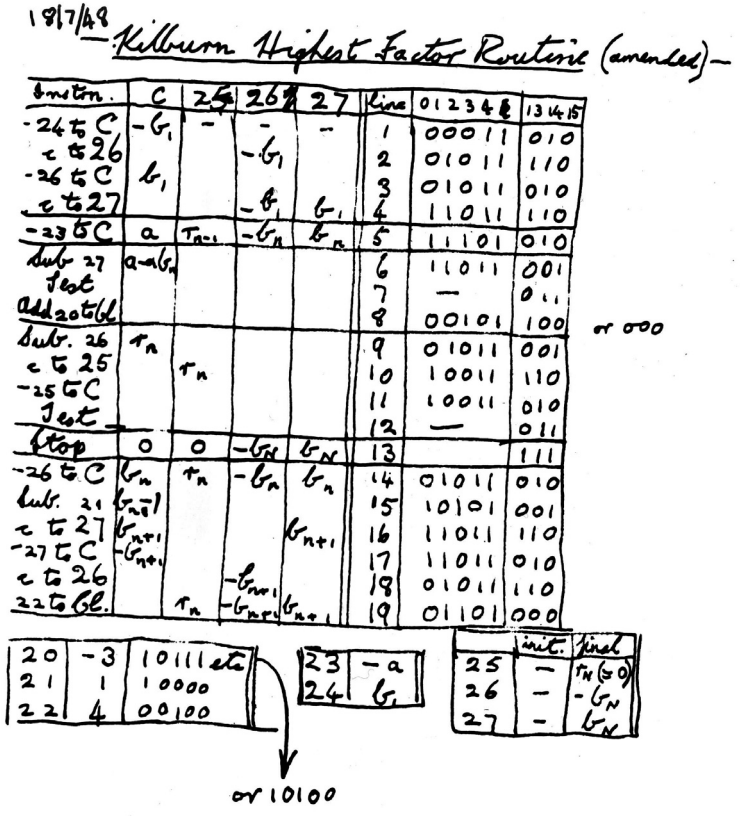


Fig. 5.7 Le programme pgd manuscrit de Kilburn.
(Departement de Computer Science, Université de Manchester, Royaume Uni)

frustrant à cette époque de faire faire un calcul par un ordinateur. Non seulement la technologie était complexe en elle-même, mais la transcription d'un algorithme sur un ordinateur en vue d'obtenir une réponse constituait une difficulté majeure.

La méthode la plus élémentaire pour le chargement du code en mémoire passait par l'utilisation d'une lignée de commutateurs électriques par lesquels les codes étaient transmis manuellement et séquentiellement dans la mémoire. Cette méthode peu efficace fut bientôt secondée par des appareils de lecture permettant de transmettre dans la mémoire de l'ordinateur les codes disposés sous la forme de cartes perforées ou de bandes de papier perforées.

Un programme en code machine était représenté par une longue liste de 0 et de 1. Pour une application typique de calcul scientifique nécessitant par exemple 200 instructions, il fallait établir une liste de 3200 bits et la transférer dans la mémoire de l'ordinateur. Ce travail manuel était d'une délicatesse extrême, car une erreur sur un seul bit réduisait le travail à néant.

Le code machine fut pendant quelque temps la forme de communication privilégiée avec les ordinateurs.

Programmer en assembleur

Une innovation salutaire intervient dans la programmation avec l'utilisation de symboles pour désigner à la fois les instructions des ordinateurs et les valeurs stockées en mémoire. Des symboles aux formes mnémoniques comme ADD, LOAD ou STORE sont ainsi définis pour désigner chacune des instructions du répertoire complet de l'ordinateur. Par ailleurs, les constantes et variables stockées en mémoire ne sont plus désignées par leur adresse de mémorisation, mais par un symbole. Cette innovation mène au programme assembleur, une forme de programme caractérisée par l'usage de symboles et une correspondance un à un entre les lignes du langage et les instructions de la machine (fig. 5.8).

```
pgd       LOAD a      ;   x=a
          STORE b     ;   b=a
boucle
          SUB b       ;   repeter x=x-b
          BLE boucle  ;   jusqu'a x<1
          BEQ fin     ;   fin si x=0
          LOAD b      ;   sinon x=b
          SUB #1      ;   x=x-1
          STORE b     ;   b=x
          JMP pdg     ;   recommencer a pgd
fin       RTS
a         DC 262144   ;   la constante a
b         DS 1        ;   la variable b
```

Fig. 5.8 Le programme pgd en assembleur.

Le programmeur écrivant maintenant son programme dans la forme symbolique prévue par le langage assembleur, cela nécessite un instrument capable de traduire son texte en code machine, compréhensible par l'ordinateur. On appelle assembleur cet outil indispensable. Pour remplir la tâche de traduction, l'assembleur procède aux phases suivantes:

1) l'exploration du texte,
2) la construction de la table des symboles,
3) la transcription des mnémoniques,
4) la génération du code machine.

Dans une publication de 1951, Maurice Wilkes introduit le terme d'assembleur pour désigner un programme qui assemble un programme principal et plusieurs sous-

routines en une seule unité d'exécution. Le nom est ensuite utilisé pour désigner un programme tel que celui reproduit à la figure 5.8, capable de traduire un langage machine symbolique (comprenant des symboles pour les instructions et les adresses) en langage machine. Parmi les premiers assembleurs citons Soap, développé pour l'IBM 650 au milieu des années 1950, et Sap développé plus tard pour l'IBM 704.

La fin des années 1950 voit arriver une nouvelle génération d'assembleurs appelés macro-assembleurs. L'article de McIlroy publié en 1960 est le premier sur ce sujet. Il concerne une extension du langage levant la contrainte de la correspondance un à un entre langage assembleur et code machine. La possibilité de générer plusieurs instructions machine à partir d'une seule instruction assembleur est évidemment un pas souhaité vers la concision du langage. Un macro-assembleur offre cette extension par la possibilité de définir des macro-instructions au moyen de macrodéfinitions et de les appeler ensuite comme de simples instructions. Il permet ainsi à un utilisateur de définir la signification de nouvelles formes du langage auxquelles il accède ensuite très simplement.

Un langage assembleur opère avec les éléments particuliers d'un processeur et dépend donc étroitement de l'architecture de ce dernier. Les processeurs ayant des architectures différentes, chacun d'entre eux nécessite son propre langage. Les langages assembleurs seront donc tous différents et incompatibles, ce qui ne facilite pas la programmation, puisqu'un programme écrit pour un processeur donné ne peut être utilisé pour un processeur d'un autre type. Pour porter un programme sur un autre processeur, il faut le réécrire dans la langue du nouveau processeur. Ce lien étroit entre langage assembleur et processeur apparaît vite comme un lourd handicap.

Vers les langages de haut niveau

Essayons de cerner les caractéristiques souhaitables d'un langage de haut niveau.

Un langage idéal est universel et valable pour l'ensemble des ordinateurs. Non seulement le langage assembleur ne remplit pas ces conditions, mais il présente aussi un niveau d'abstraction beaucoup trop faible. Un langage de haut niveau devrait être en effet capable de concrétiser des concepts plus évolués, comme des fonctions complexes ou des matrices, et il devrait présenter les caractéristiques suivantes:

- une indépendance vis-à-vis de la machine,
- une intelligibilité plus grande,
- une capacité de suppression de fautes,
- l'efficacité du programmeur.

Les années 1950 sont marquées par cette recherche d'un langage de haut niveau [1, 7]. C'est l'époque des pionniers du langage de programmation [3].

Plankalkül (1946)

Vers la fin de la guerre, l'ingénieur allemand Konrad Zuse travaille seul, réfugié dans les Alpes bavaroises. Manquant de moyens pour poursuivre le travail sur son

ordinateur à relais Z4, il conclut dès 1945 son travail sur la notation formelle pour les algorithmes. Ses efforts débouchent sur le langage Plankalkül.

Plankalkül est conceptuellement très avancé. Zuse ne s'est pas limité aux capacités de sa propre machine, mais il a aussi inclus dans son langage des instructions plus générales, notamment une instruction d'itération. La contribution la plus importante réside dans un riche répertoire de types de données. Il est notamment possible de composer d'autres types à partir d'un type bit de base. Il y a des types entier, flottant, des tableaux de dimension arbitraire, des structures de données qui permettent de combiner des types de données arbitraires. Dix ans passent avant que n'apparaisse une notion aussi compréhensive des structures de données dans un autre langage de programmation. Malgré différentes tentatives entreprises par Zuse, Plankalkül n'a jamais été implémenté. Il aura surtout servi d'aide conceptuelle à la programmation.

Zuse a écrit et publié un grand nombre de programmes dans ce langage: algorithmes de tri, de test sur la connectivité de graphes ou pour le jeu d'échec. Il a ainsi apporté une contribution fondamentale et pratique à la programmation.

Tout le travail de Zuse est cependant resté largement inconnu. Le manuscrit qui décrit le langage Plankalkül ne sera pas publié avant 1972. La programmation naissante, en particulier outre-Atlantique, ne profitera pas de ces inventions remarquables.

Short Code (1949)

Short Code est le premier langage de programmation à avoir été implémenté. On le doit à Mauchly, un des inventeurs de l'ENIAC, qui le conçoit en 1949. Le langage est conçu pour aider les programmeurs à entrer des expressions algébriques comme $A = B + (C \times D)$ dans un ordinateur. En raison de l'architecture à 12 caractères des mots de la machine Univac à laquelle ce langage était destiné, il fut convenu de prendre deux caractères pour coder chaque variable et chaque opérateur. Une expression Short Code est donc codée en groupes de 6×2 caractères. Par exemple, l'expression $X = Y - Z$, qui nécessite trois variables Vi et deux opérateurs, est représentée comme suit:

$X = Y - Z \quad \Rightarrow \quad$ 00 V0 03 V1 01 V2

Short Code est bien sûr assez primitif, mais représente cependant une amélioration notable en comparaison du code machine.

Le codage automatique

Au début des années 1950, plusieurs équipes américaines et européennes travaillent sur le problème de la programmation, désignée à l'époque sous le nom de «codage automatique». L'objectif de ces travaux est de produire le code des ordinateurs de manière spontanée, c'est-à-dire d'inventer le premier véritable langage de programmation.

Professeur à l'Ecole polytechnique fédérale (ETH) de Zurich, Heinz Rutishauser développe en 1951 des méthodes simples de génération de code. Un étudiant anglais,

Alick E. Glennie, écrit l'année suivante le système Autocode pour le Mark I de Manchester. Ce langage est considéré comme l'un des premiers implémentés sur un vrai ordinateur avec une réelle utilité. Il compile toutefois un langage assembleur symbolique, bas niveau et orienté vers une seule machine.

Au Etats-Unis, Grace Hopper est une pionnière de la programmation automatique. En 1952, alors qu'elle travaille pour Remington Rand Univac, elle introduit A-0, le premier d'une série de systèmes de programmation à exploiter la notion de macro-expansion. Le langage commercial qui en résulte est vendu dès 1957 par la société sous le nom de Math-Matic. Outre ces travaux, Grace Hopper se manifeste comme une fervente adepte de la programmation automatique. C'est lors d'un symposium qu'elle organise en 1954 que le système de traduction algébrique, écrit en mai 1953 par Laning et Zierler pour le Whirlwind du MIT, gagne sa notoriété. Ce système est généralement considéré comme le premier vrai langage compilé.

Cet enthousiasme pour le codage automatique n'est toutefois de loin pas partagé par tous. La communauté des programmeurs est lente à réaliser le potentiel offert par les langages pour la programmation des machines. Nombreux sont ceux qui pensent que les avantages de la programmation en code machine – notamment la grande rapidité d'exécution et la bonne compacité de code – valent l'effort supplémentaire de programmation.

La petite histoire veut que c'est Grace Hopper qui soit à l'origine du terme de «bogue informatique». Alors qu'elle travaille sur un prototype de Mark II au cours de l'été 1945, elle découvre qu'un papillon de nuit coincé dans un relais de l'ordinateur est à l'origine d'une erreur de calcul. On désigne depuis lors une faute dans un programme informatique sous le terme de *bug* (insecte, en américain). Le nom de «debugger» a par ailleurs été donné aux programmes spécialisés dans la recherche de fautes. Les termes francisés de «bogue» et de «débogueur» ne se sont jamais véritablement imposés, peut-être en raison de la force de l'histoire!

FORTRAN (1954-1958)

FORTRAN n'est pas le premier langage de programmation. C'est cependant de loin le premier langage compilé à être accepté et utilisé à large échelle. C'est aussi le premier langage à être commercialisé.

FORTRAN est conçu par une équipe d'IBM dirigée par John Backus. Il est très probablement inspiré des travaux du MIT, Backus ayant notamment rendu visite à Laning et Zierler après le symposium de 1954. Si l'impact technique des travaux du MIT sur les spécifications de FORTRAN demeure peu évident, ces contacts ont probablement contribué à soutenir l'équipe dans sa quête d'un langage à la fois efficace et facile à coder. Backus rapporte qu'à cette époque, l'inquiétude majeure était que le compilateur ne soit pas assez performant et aboutisse à une application dont le code généré serait deux fois plus lent que si elle avait été codée à la main. Si une telle éventualité s'était produite, elle aurait donné gain de cause aux sceptiques et compromis l'acceptation du nouveau système.

```
C       CALCULE LA NORME D'UN VECTEUR
        DIMENSION V(5)
1       FORMAT (5F8.4)
        READ 1,V
        DI = 0.0
        DO 10 I=1,5
        DI=DI+V(I)**2
        IF ACCUMULATOR OVERFLOW 100,10
10      CONTINUE
        DI=SQRTF(DI)
        PRINT 2,DI
2       FORMAT (10H DISTANCE, F12.4)
        GO TO 200
100     PRINT 3
3       FORMAT (16H VECTOR TOO LONG)
200     STOP
```

Fig. 5.9 Exemple FORTRAN.

Ce souci d'efficacité a notamment pour conséquence d'offrir un premier compilateur produisant – très lentement – du code efficace: le compilateur FORTRAN pour l'IBM 704, distribué à partir d'avril 1947, avait besoin d'une minute pour compiler la seule instruction "STOP"! Mais à l'époque, les préoccupations s'attachaient plus aux performances d'exécution que de compilation: cette dernière ne devait pas être nécessairement rapide, puisqu'elle n'était pas exécutée fréquemment. Ce n'est que par la suite qu'une compilation rapide est devenue un caractère important du langage.

FORTRAN signifie *FORmula TRANslator*. Il est avant tout destiné à la programmation scientifique, raison pour laquelle il comprend dès le départ des nombres à virgule flottante ainsi que l'indexage de tables (fig. 5.9).

Il existe différentes versions de FORTRAN. La première, FORTRAN I, est lancée en avril 1957, avec les caractéristiques suivantes:

- assignations, variables, expressions,
- nombres entiers, nombres flottants,
- tableaux de taille fixe,
- DO, IF, GOTO,
- entrée/sortie par FORMAT,
- types implicites (I, J, K, L, M, N).

FORTRAN II voit le jour en 1958 et introduit notamment la compilation séparée des sous-routines. Cette possibilité de compiler séparément les différentes parties du programme était nécessaire à cause des aléas de fonctionnement du matériel. Les programmeurs étaient souvent interrompus par des pannes de matériel et il leur était difficile de compiler des programmes plus longs que 300 à 400 lignes.

FORTRAN est au départ conçu pour l'IBM 704 et n'a pas la prétention d'être indépendant de la machine. FORTRAN I et FORTRAN II comprennent des instructions qui dépendent de la machine, mais ceci change avec l'apparition de FORTRAN IV en 1962. Cette version introduit aussi des types de variables nouveaux: booléen, double précision, complexe, réel, entier et externe.

FORTRAN devient peu à peu plus universel et se dissémine rapidement. Chaque constructeur d'ordinateur produit ou s'engage à offrir un compilateur FORTRAN. Ce langage devient le plus utilisé dans la communauté scientifique jusqu'aux alentours de 1978, et demeure toujours très utilisé de nos jours, notamment en programmation mathématique numérique et scientifique.

FORTRAN 77 est lancé en 1977 mais ne constitue qu'une révision assez modeste au vu des connaissances de l'époque. Il apporte des outils de manipulation de chaînes et une instruction IF... ELSE ... La dernière version, FORTRAN 90, offre des tableaux dynamiques, des pointeurs et la possibilité d'effectuer des opérations sur les tableaux.

John Backus

Sa maîtrise de mathématique en poche, John Backus entre en 1950 chez IBM, comme programmeur du SSEC. Il devient ensuite responsable d'un système interpréteur nommé Speedcoding et destiné à l'IBM 701. Entre 1954 et 1958, il dirige le groupe de recherche en programmation puis développe FORTRAN. Devenu membre des commissions qui définissent ALGOL 58 et ALGOL 60, il entre chez IBM Research. Afin de formaliser ALGOL 58, il utilise la technique de description syntaxique qui porte maintenant son nom, la forme de Backus et Naur (BNF). Backus s'intéressera par la suite au style de programmation fonctionnelle.

Evolution des langages et des paradigmes

A peine conçus, les langages de programmation évoluent rapidement au gré des besoins et des performances accrues des ordinateurs.

Après FORTRAN germent, éclosent, s'épanouissent et meurent de nombreux autres langages. Mais au jardin des langages de programmation, aucune espèce ne parvient à s'imposer véritablement. Leur évolution est dominée par l'apparition de nouvelles espèces sachant mettre à profit des qualités héritées de leurs parents.

La figure 5.10 situe quelques langages de programmation importants selon leur apparition dans le temps et le paradigme dont ils relèvent.

Paradigme des langages impératifs

Les premiers langages sont dits impératifs. Selon ce paradigme, le programme dicte la succession précise des opérations que l'ordinateur doit impérativement exécuter.

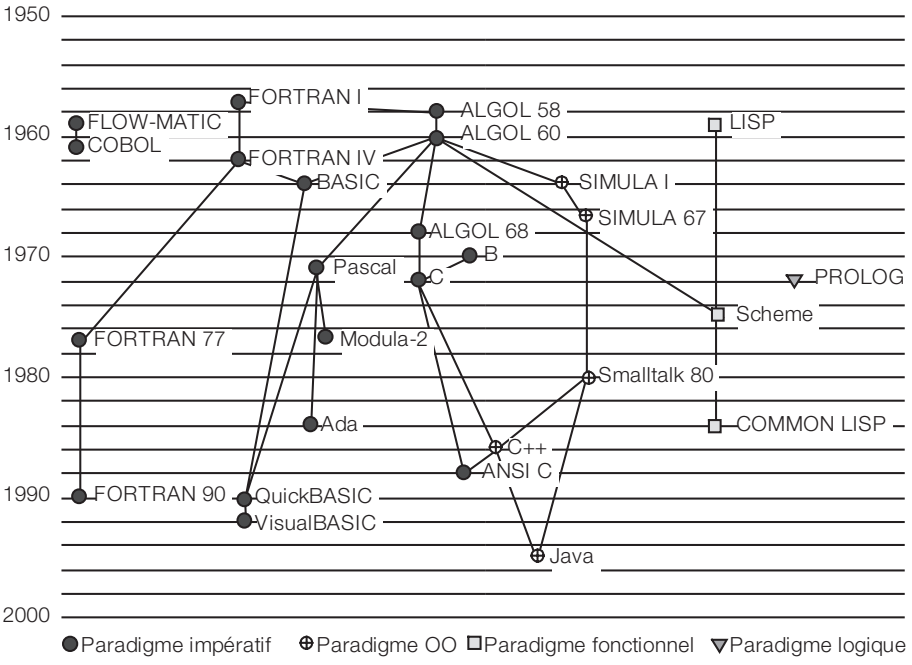


Fig. 5.10 Evolution des langages et des paradigmes.

Lors de la création des premiers langages, les concepteurs sont dominés par l'idée d'imiter et de rendre plus abstraites les opérations à effectuer sur un ordinateur. La forme des premiers ordinateurs a donc une influence sur cette abstraction, et le modèle d'ordinateur de Von Neumann inspire le paradigme des premiers langages. Le paradigme impératif est basé sur:

- une exécution séquentielle des opérations,
- un usage de variables pour désigner des places en mémoire et l'assignation pour changer leur valeur.

Le terme «impératif» se rapporte au contrôle de l'exécution séquentielle des opérations et indique clairement que c'est le programme qui dicte l'ordre dans lequel s'exécutent les différentes instructions de la machine.

Le paradigme impératif est à la base de langages de programmation comme FORTRAN, PASCAL, BASIC et C, et reste présent dans de nombreux langages utilisés aujourd'hui.

Avec le développement de la programmation, il apparaît que d'autres manières de décrire des calculs existent, et que le paradigme impératif n'est qu'un paradigme de calcul parmi d'autres, comme le paradigme fonctionnel et le paradigme logique.

ALGOL 60 (1957-1960)

Conçu par les meilleurs experts comme le langage de programmation universel, ALGOL ne parvint pas à s'imposer.

L'idée de disposer d'un langage de programmation universel, ouvert à tous les pays et toutes les machines, était attrayante. FORTRAN ne pouvait être utilisé ainsi car il appartenait à IBM. Une commission d'experts se réunit donc en 1958 afin de concevoir un nouveau langage. Le résultat de ces travaux est ALGOL 58 (ALGOrithmic Language), un langage plus général que FORTRAN, et le premier conçu pour être totalement indépendant de toute machine (fig. 5.11).

Mais les chercheurs ayant décidé de ne pas inclure d'instructions d'entrée/sortie, le portage de programmes s'avérait extrêmement compliqué à réaliser. Satisfait du succès de FORTRAN, IBM n'offrait par ailleurs pas de support ALGOL sur ses machines. Ces deux éléments empêchèrent définitivement ALGOL de devenir populaire.

L'importance d'ALGOL repose sur les nouveaux concepts que ce langage a eu le mérite d'introduire:

- structure de blocs,
- routines récursives,
- structure de bloc avec visibilité,
- IF..THEN...ELSE emboîtés.

Chacun de ces concepts est important. Le bloc, qui permet de structurer un programme en unités bien définies, porte en lui la clé de la modularité nécessaire à la construction de grands ensembles de programmes. La plupart des langages modernes ont repris à leur compte les nouveaux concepts implantés dans ALGOL.

Ecarté de la sphère commerciale, ALGOL s'est surtout développé dans la sphère académique, et est devenu une base d'apprentissage des concepts de la programmation pour les étudiants. Il est par ailleurs demeuré pendant 20 ans le langage principal utilisé pour décrire et publier des algorithmes.

```
integer procedure combinations(n,m);
  integer n,m;
  begin
    integer procedure factorial (n);
          value n; integer n;
          factorial:= if n>0 then n*factorial(n-1) else 1;
    combinations:= factorial(n) / (factorial(m)*factorial(n-m))
  end
```

Fig. 5.11 Exemple d'ALGOL.

La réunion en 1958 de la commission qui a conçu ALGOL a aussi une importance historique, puisqu'elle marque la rencontre de deux courants du développement des langages algorithmiques, l'américain et l'européen. Le choix de Zurich comme lieu de réunion et la composition de la commission – quatre Américains et quatre Européens – sont à cet égard significatifs. Si l'existence du courant américain est évidente, celle du courant européen l'est beaucoup moins. Il était notamment porté par Heinz Rutishauser, qui après avoir poursuivi des travaux sur la Z4 de Zuse, consacra une grande partie de son énergie au développement d'ALGOL. Ce langage est demeuré le favori de l'ETHZ jusqu'en 1970, année où il est détrôné par le langage Pascal que vient d'inventer le professeur N. Wirth.

Le formalisme de Backus-Naur (BNF)

Le langage FORTRAN est dépourvu de structure formellement claire. Ainsi les instructions

```
DO 99 I = 1,10
DO 99 I = 1.10
```

ont des interprétations totalement différentes. Si la première instruction est la forme familière de la boucle DO, la seconde assigne la valeur 1,1 à la variable DO99I. Cet état de fait rend difficile l'écriture de programmes dépourvus de fautes.

Une amélioration décisive des langages apparaît avec la suggestion faite par John Backus lors de la définition d'ALGOL 58: il propose de donner une définition claire des unités lexicales élémentaires du langage, comme les séparateurs, les identificateurs, les constantes, les mots clés, les opérateurs, puis de définir une grammaire dictant formellement les manières possibles d'écrire des phrases correctes avec les unités lexicales de ce langage. Sur la base de connaissances glanées dans un cours sur la logique, Backus propose l'usage d'une grammaire non contextuelle reposant sur des productions caractérisées par l'usage des métasymboles::=, |, < ou >.

☞ Pour définir une expression, on commence donc par définir les chiffres, sachant qu'un nombre est fait de chiffres et qu'une expression comprend des nombres combinés avec des opérateurs + et × ainsi qu'éventuellement des parenthèses. En notation BNF:

<digit>::= 0|1|2|3|4|5|6|7|8|9
<number>::=<number><digit>|<digit>
<exp>::=<exp>+<exp>|<exp> × <exp>|(<exp>)|<number>

L'application systématique de cette description permet de décrire la structure de tout le programme et de toutes les unités le composant. Ainsi la description de Pascal:

<program>::= program <...><...>.

signifie qu'un programme Pascal commence par le mot «program», qu'il comprend d'autres éléments et qu'il se termine par un point.
Quant à la désignation BNF, la signification originale de Backus Normal Form s'est peu à peu transformée en Backus-Naur Form suite à la contribution de Naur.

Notons que Niklaus Wirth a fait largement usage de la forme syntaxique pour décrire le langage Pascal. Il a notamment utilisé une forme étendue de BNF nommée EBNF, comprenant des métasymboles supplémentaires {...} utilisés pour signifier zéro ou plus de répétitions de l'élément mis entre parenthèses. Une simplification de l'écriture en résulte. L'exemple:

<number>::=<digit>{<digit>}

signifie que <number> consiste en un ou plusieurs <digit>.

Le fait qu'une telle grammaire puisse aussi se représenter sous la forme de diagramme syntaxique a été largement utilisé dans les manuels Pascal pour décrire graphiquement la grammaire du langage. Les diagrammes syntaxiques de la figure 5.12 sont équivalents aux formes EBNF données dans le texte, et en dérivent directement.

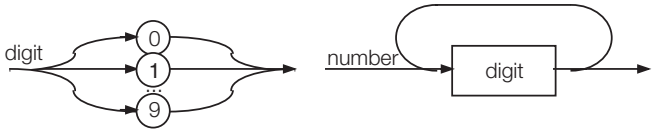


Fig. 5.12 Diagrammes syntaxiques.

COBOL 61 (1959-1961)

Critiqué dès le départ pour sa forme verbeuse, COBOL a le privilège de figurer parmi les langages les plus utilisés.

Comme l'indique son nom (*COmmon Business-Oriented Language*), le langage COBOL est destiné à couvrir les besoins de traitement des données dans les applications commerciales. Ses concepteurs visent la performance et la simplicité, afin d'obtenir un langage à la fois accessible au programmeur novice et compréhensible par les directeurs. L'un de ses précurseurs est le langage FLOW-MATIC, développé dans les années 1950 par Grace Hopper. D'autres projets sont lancés et aboutissent finalement au langage COBOL (fig. 5.13).

Une première caractéristique du langage est sa forme verbeuse, objet de nombreuses critiques. Ainsi l'expression FORTRAN a = b + c s'écrit en COBOL

ADD B TO C GIVING A

Ceci n'empêche pas COBOL de devenir un langage majeur grâce à ses nombreuses qualités. COBOL innove par rapport à FORTRAN avec les éléments suivants:

- une description de données indépendante de la machine,
- des définitions de données récursives,
- une structure conditionnelle puissante (IF-THEN-ELSE),
- une séparation claire des instructions et des données,

- un langage indépendant de la machine, basé sur des possibilités matérielles réalistes.

COBOL est une histoire à succès et se répand rapidement. Il est choisi comme langage pour la Défense américaine et c'est l'un des premiers langages à être standardisé. L'engouement pour ce langage est lié aux moyens performants de description des données dont il est doté, mais aussi à la garantie de pérennité que lui confère une bonne standardisation. Ces qualités ont notamment prévalu sur les faiblesses et imperfections des moyens de traitement. Ce langage, le plus utilisé dans le passé, demeure encore en usage de nos jours.

```
DATA DIVISION.
WORKING STORAGE SECTION.
01 NUMERIC VARIABLE USAGE IS COMPUTATIONAL.
  02 TERMS PICTURE 9999 OCCURS 100 TIMES INDEXED BY I.
  02 N-PICTURE 999.
  02 SUM PICTURE 999999.
  02 HALF-TERM PICTURE 9999.
  02 RMDR PICTURE 9.

PROCEDURE DIVISION.
EXAMPLE.
  MOVE 23 TO TERMS(1).
  MOVE 34 TO TERMS(2).
  MOVE 7 TO TERMS(3).
  MOVE 9 TO TERMS(4).
  MOVE 4 TO N.
  PERFORM SUM-ODDS.
SUM-ODDS.
  MOVE 0 TO SUM.
  PERFORM CONSIDER-ONE-TERM VARYING I FROM 1 BY 1 UNTIL I>N
CONSIDER-ONE-TERM.
  DIVIDE 2 INTO TERMS(I) GIVING HALF-TERM REMAINDER RMDR.
  IF RMDR IS EQUAL TO 1 ADD TERMS(I) TO SUM.
```

Fig. 5.13 Exemple COBOL.

Quant aux objectifs de simplicité poursuivis par ses concepteurs, force est de constater que programmer en COBOL se révèle aussi difficile qu'avec tout autre langage, malgré les efforts déployés pour imiter le langage naturel. C'est à la construction même des algorithmes et non plus à la description des détails du programme que les chercheurs vont s'attacher pour tenter de simplifier la programmation des ordinateurs.

LISP (1959-1960)

Conçu à la même époque que FORTRAN et ALGOL, le langage LISP a survécu jusqu'à nos jours sans jamais entrer en concurrence avec ses contemporains.

Développé par John McCarthy entre 1956 et 1958 et implémenté pour la première fois entre 1959 et 1962, LISP (LISt Processing) est conçu comme le langage de l'intelligence artificielle. Utilisé au départ pour des programmes qui calculent symboliquement des intégrales et des dérivées ainsi que d'autres fonctions destinées à la vérification de théorèmes, il est ensuite employé dans l'un des premiers jeux interactifs d'action et de combat, Abuse. LISP est à la base du développement de systèmes expérimentaux dans de nombreux domaines de l'intelligence artificielle, telle que l'interprétation du langage et les systèmes experts.

```
(DEFUN SUMODDS
   (LAMBDA (TERMS)
      (COND
         ((NULL TERMS) 0)
         ((ODD (CAR TERMS)) (PLUS(CAR TERMS) (SUMODDS (CDR TERMS))))
         (T (SUMODDS (CDR TERMS)))))))

(SUMODDS '(23 34 7 9))

(SUMODDS'(23 34 7 9))
  = (PLUS 23 (SUMODDS'(34 7 9))
    = (PLUS 23 (SUMODDS'(7 9))
      = (PLUS 23 (PLUS 7 (SUMODDS'(9)))
        = (PLUS 23 (PLUS 7 (PLUS 9 (SUMODDS'( )))))
      = (PLUS 23 (PLUS 7 (PLUS 9 0)))
    = (PLUS 23 (PLUS 7 9))
  = (PLUS 23 16)
= 39
```

Fig. 5.14 Exemple LISP.

A la différence de ses concurrents, LISP est avant tout un langage fonctionnel. Il est le premier à offrir la récursion, des fonctions de première classe, la *garbage collection* (qui consiste à récupérer automatiquement les emplacements en mémoire qui ne sont plus utilisés par le système) ainsi qu'une définition formelle du langage. LISP est unique dans sa capacité à exprimer des algorithmes récursifs qui manipulent des structures de données dynamiques. Il présente une forme inhabituelle et une grande simplicité de style (fig. 5.14).

Formellement, tous les éléments de LISP s'écrivent comme une suite de mots encadrée par des parenthèses. Ses détracteurs, irrités par l'abondance des parenthèses

requises dans le texte des programmes, ont d'ailleurs prétendu que LISP signifiait «*Lots of Irritating Superfluous Parentheses*».

En évoluant, le langage LISP donne naissance à un grand nombre de dialectes qui finissent par rendre difficile la communication dans ce langage. Un effort d'uniformisation entrepris par une commission au début des années 1980 a conduit au langage Common LISP.

John McCarthy

Docteur en mathématiques (1951), John McCarthy enseigne tout d'abord les mathématiques à Princeton puis à Dartmouth. Il participe aux développements d'ALGOL 58 et 60, avant d'entrer au MIT, où, avec Marvin Minsky, il organise et dirige le projet d'intelligence artificielle. C'est là qu'il crée le langage LISP.

Paradigme des langages fonctionnels

S'il paraît logique de formuler des opérations en termes de fonctions, il est plus difficile de concevoir que tout peut être programmé à partir de fonctions.

Le paradigme fonctionnel est issu des mathématiques et repose sur l'évaluation de fonction, c'est-à-dire l'application de fonctions à des valeurs connues. Le principe considère que les solutions du problème sont obtenues par stricte application des fonctions prévues aux valeurs initiales connues que sont les données du problème. Ce principe se passe de la notion de variable ou de l'assignation pour changer sa valeur. De même, les opérations répétitives ne sont pas exprimées par des boucles, mais par des fonctions récursives. Ce paradigme représente une vue sensiblement différente de la programmation impérative. La programmation fonctionnelle n'emploie ni variables ni boucles [8] et elle est extrêmement rigoureuse. Traités comme des données, les programmes offrent une grande flexibilité, notamment pour le traitement de données symboliques. Des représentants typiques sont Lisp, Scheme et Mathematica.

Le désavantage traditionnel des langages fonctionnels est longtemps demeuré l'inefficacité d'exécution du code généré. Toutefois, un regain d'intérêt pour le paradigme fonctionnel s'est manifesté récemment suite à l'apparition de nouveaux compilateurs plus performants, et avec la possibilité offerte de calculer plus rapidement en passant sur des architectures parallèles. Le paradigme fonctionnel est en effet libéré de l'exécution strictement séquentielle caractéristique des langages impératifs et laisse ouverte la possibilité du calcul parallèle, comme l'évaluation simultanée de plusieurs fonctions.

BASIC (1964)

Depuis son apparition, la programmation demeurait réservée aux spécialistes. Deux professeurs remédient à cet état de fait en développant un langage simple et accessible à tous: le BASIC.

Conçu au départ pour la machine General Electric 225, le langage BASIC est développé en 1964 par John Kemeny et Thomas Kurtz, professeurs au Dartmouth College dans le New Hampshire. Le collège est à l'époque au centre du développement d'un système d'exploitation temps réel et le nouveau langage doit supporter la communication avec l'utilisateur. Deux principes dictent sa réalisation:

- Le système doit être simple et facile à utiliser par un programmeur occasionnel.
- Quand le choix existe entre simplicité et performance il faut choisir la simplicité.

```
100 DIM T(100)
200 READ N
300 FOR I=1 TO N
400 READ T(I)
500 NEXT I
600 GOSUB 1100
700 PRINT S
800 GOTO 2000
900 DATA 4
1000 DATA 23,34,7,9
1100 REM MAKE S THE SUM OF THE ODD ELEMENTS IN ARRAY T(1..N)
1200 LET S = 0
1300 FOR I = 1 TO N
1400  IF NOT ODD(T(I)) THEN GOTO 1600
1500  LET S = S + T(I)
1600 NEXT I
1700 RETURN
2000 END
```

Fig. 5.15 Exemple BASIC.

Insatisfaits par la complexité des langages de programmation de l'époque (FORTRAN et ALGOL) et du passage obligé par cartes perforées pour introduire un programme en mémoire, Kemeny et Kurtz cherchent à développer un langage simple, facile à utiliser et à apprendre (fig. 5.15). Dartmouth étant essentiellement une école d'art, ses élèves utilisent difficilement les langages de l'époque, malgré leurs connaissances en mathématiques. Les deux enseignants sont convaincus qu'il faut trouver une solution pour combler le fossé entre professionnels et non spécialistes de la programmation. L'objectif de simplicité de BASIC transparaît dans l'acronyme qui lui tient lieu de nom (Beginner's All-purpose Symbolic Instruction Code). Il devient vite très populaire et s'impose partout où des exigences de simplicité prédominent.

Compilateur et interpréteur

Les langages de programmation requièrent des compilateurs et des interpréteurs de la même manière que les langages naturels emploient des traducteurs et des interprètes.

Un langage est un ensemble de règles de grammaire. Pour programmer, il faut disposer de l'implémentation du langage, c'est-à-dire de l'outil spécifique capable de traduire en code exécutable un programme écrit selon ces règles. Le compilateur est la forme privilégiée de cette implémentation. Il lit le texte du programme et le traduit en code machine, qui peut ensuite être exécuté sur la machine indépendamment du compilateur (fig. 5.16). Il revient à Grace Hopper d'avoir introduit le terme de compilateur en 1951 pour désigner ce processus de traduction de programme au cours duquel le programme extrait ou compile des séquences d'instructions machine.

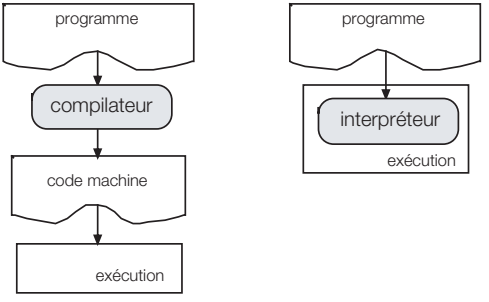


Fig. 5.16 Traduction par un compilateur ou un interpréteur.

L'autre forme d'implémentation se nomme interpréteur. A la différence du compilateur qui traduit tout le texte d'un programme de haut niveau d'un seul coup, l'interpréteur transforme ce texte ligne par ligne et exécute immédiatement le code machine correspondant. Ce processus convient particulièrement aux situations dans lesquelles l'utilisateur souhaite une exécution au fur et à mesure de l'introduction des lignes de son texte, comme c'est le cas avec des systèmes interactifs.

On trouve une situation comparable dans la traduction des langages naturels comme le français ou l'allemand. Le traducteur en allemand se rapproche du compilateur BASIC car il fournit une traduction portant sur l'ensemble d'un texte. L'interprète en allemand est comparable à l'interpréteur BASIC car il traduit le langage au fur et à mesure.

La compilation offre une indépendance et une rapidité d'exécution du code généré, alors que l'interprétation propose une interactivité pendant l'écriture même du programme. Bien que tout langage puisse être théoriquement compilé ou interprété – moyennant des initialisations nécessaires – les langages comme FORTRAN et COBOL sont généralement compilés alors que Forth, Logo et APL sont presque toujours interprétés. BASIC et LISP sont disponibles dans les deux formes.

La programmation structurée et le GOTO

Un tournant évolutif apparaît lorsque la programmation structurée se voit érigée au rang de principe sacré et que le GOTO devient sacrilège.

Le terme de programmation structurée est introduit par Dijkstra en 1969 dans une publication intitulée «Notes on structural programming». L'article met l'accent sur l'importance du style de programmation et sur la qualité des programmes. Il observe que «la qualité des programmeurs est inversement proportionnelle à la densité des instructions GOTO trouvées dans leurs programmes». Cette phrase a un effet explosif et lance le long débat du rôle du GOTO en programmation.

Au sens strict, la programmation structurée concerne le développement de programmes avec un ensemble d'instructions petit ou suffisant, en particulier pour les instructions de contrôle. Au sens large, elle présuppose des objectifs comme la simplicité, l'intelligibilité, la vérifiabilité, la modificabilité et la maintenance des programmes [18].

Le débat est notamment alimenté par les fondamentalistes qui s'appuient sur une idée forte, le théorème de Böhm et Jacopini (publié en 1966), permettant la structuration suivante: l'expression de tout programme avec les constructeurs de séquence (<>;<>), d'alternative (IF <> THEN <> ELSE<>) et d'itération (WHILE <> DO <>). Le GOTO peut donc être supprimé de tout programme grâce à l'emploi de ces constructeurs (fig. 5.17).

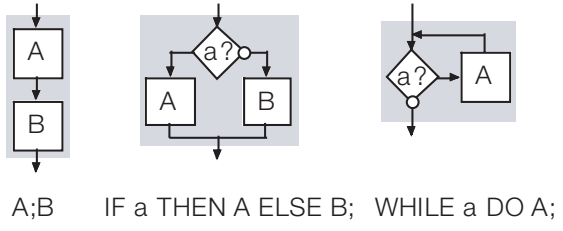


Fig. 5.17 Trois constructeurs formant un ensemble complet.

Les défenseurs du GOTO font remarquer que sa suppression systématique, bien que possible en principe, n'est pas souhaitable lorsqu'elle rend le programme plus difficile à lire et à comprendre, puisque ceci va à l'encontre même de l'objectif de la programmation structurée. Le cas typique est la détection dans un programme des erreurs exigeant un traitement d'exception, qui est particulièrement lourde à réaliser sous la forme structurée stricte. Les deux camps finissent par s'entendre sur la nécessité de pratiquer la programmation structurée dans un esprit de clarification des programmes, dans lesquels le GOTO doit être évité de préférence, tout en demeurant acceptable [2].

ALGOL 68 (1963-1969)

ALGOL 68 est développé comme une version évoluée d'ALGOL 60. L'une des nouveautés proposées par ce langage est la possibilité de construire des types composites à partir de types de base simples comme *int*, *real*, *bool*, *char*, *format* ainsi que d'opérateurs de construction tels que «multiple» *[ ]m*, «structure» *struct(m, n)* et «référence» *ref m*. ALGOL 68 permet donc de construire les compositions les plus variées sous une forme systématique, comme:

struct([ ]ref m, [ ]struct (m, ref ref n))

Cette nouvelle mouture du langage fait l'objet du rapport ALGOL 68, rédigé par un groupe de concepteurs. Sa lecture se révèle d'une extrême difficulté, notamment à cause de sa vaste formalisation. Le manque d'implémentations d'ALGOL 68 ainsi que de manuels d'utilisateur participent aussi au rejet dont il fait l'objet de la part des programmeurs. Trop général et trop flexible pour être assimilé et utilisé par eux, ce langage ne parviendra jamais à s'imposer.

Pascal (1971)

Où l'on voit un déçu d'ALGOL créer un langage à succès, simple et très formel.

Le langage Pascal tire son nom de celui du célèbre philosophe et mathématicien français du XVII^{e} siècle, et il est développé à la fin des années 1960 par Niklaus Wirth. Son enthousiasme d'enfant pour les avions radioguidés le conduit à étudier l'électronique; il obtient les diplômes d'ingénieur de l'ETH de Zurich ainsi que de l'Université de Laval au Québec [13]. Après un doctorat à Berkeley, il commence à s'intéresser aux langages de programmation et devient professeur assistant à Stanford. Il participe à l'élaboration d'ALGOL 68, mais considère ces développements comme trop généraux. De retour en Suisse, il s'attache à la conception d'un langage privilégiant la simplicité, tant au niveau de la conception que de l'utilisation et de l'implémentation. Le langage Pascal qu'il met au point introduit peu de nouveautés, mais réunit les meilleures caractéristiques des langages existants (fig. 5.18). Pascal est conçu comme un outil d'enseignement et c'est à cette fin qu'il est tout d'abord utilisé. En raison de sa grande simplicité, ce langage ne supporte pas certaines caractéristiques pourtant considérées comme essentielles par les programmeurs, telle que la compilation séparée. Malgré ces inconvénients, il devient vite très populaire grâce à sa simplicité, sa puissance, sa convivialité et la facilité avec laquelle il peut être maîtrisé. Wirth créera plus tard les langages Modula-2 et Oberon.

Prolog (1970)

Dix ans après l'arrivée des langages impératifs et fonctionnels, le langage Prolog propose des principes nouveaux.

```
program SumOddNumbers;
type        TermIndex = 1..100;
            TermArray = array[TermIndex] of integer;
var         myTerms: TermArray;
function SumOdds(n: TermIndex; terms: TermArray): integer;
   var i: termIndex;
   sum: integer;
   begin
      sum:= 0;
      for i:=1 to n do
         if Odd(terms[i]) then
           sum:= sum + terms[i];
      SumOdds :=sum;
   end;
begin
   myTerms[1]:=23; myTerms[2]:=34; myTerms[3]:=7; myTerms[4]:=9;
   Writeln(SumOdds(4,myTerms))
end.
```

Fig. 5.18 Exemple Pascal.

Prolog voit le jour à l'Université de Marseille, suite aux travaux d'un groupe de chercheurs mené par le jeune professeur Alain Colmerauer. Son nom est une abréviation de PROgrammation LOGique, l'outil logiciel de communication homme-machine développé par l'équipe. L'idée poursuivie depuis quelques années par Alain Colmerauer est d'établir une communication avec l'ordinateur au moyen d'un langage naturel comme le français, projet considéré à l'époque comme relativement utopique par ses pairs. Le premier système Prolog est réalisé fin 1972 et sert à l'implémentation d'un système de communication homme-machine pour la langue française. Ce système est en mesure d'analyser les noms communs au singulier et au pluriel, ainsi que les verbes conjugués, même irréguliers. Voici un exemple de dialogue:

Introduction
TOUT PSYCHIATRE EST UNE PERSONNE.
CHAQUE PERSONNE QU'IL ANALYSE, EST MALADE.
*JACQUES EST UN PSYCHIATRE A *MARSEILLE.
EST-CE QUE *JACQUES EST UNE PERSONNE?
OU EST *JACQUES?
EST-CE QUE *JACQUES EST MALADE?
Réponses
OUI. A MARSEILLE. JE NE SAIS PAS.

Contrairement aux Etats-Unis, l'Europe adopte rapidement Prolog comme outil de programmation. Au Japon, ce langage est au cœur du développement des ordinateurs

de la cinquième génération. Les qualités de Prolog lui permettent de devenir un des langages majeurs utilisés en intelligence artificielle et pour les applications non numériques (fig. 5.19).

```
add (Alain est-parent-de Claude)
add (Alain est-parent-de Daniel)
add (Bea est-parent-de Claude)
add (Bea est-parent-de Daniel)
add (Claude est-parent-de Eric)

which (x: x est-parent-de Daniel)
Alain
Bea
No (more) answers

which (x: Bea est-parent-de x)
Claude
Daniel
No (more) answers

add (x est-ancetre-de y if x est-parent-de y)
add (x est-ancetre-de y if z est-parent-de y and x est-ancetre-de z

which (x: x est-ancetre-de Eric)
Claude
Daniel
Bea
No (more) answers

which (x: Alain est-ancetre-de x)
Claude
Daniel
Eric
No (more) answers
```

Fig. 5.19 Exemple Prolog.

Paradigme des langages logiques

La logique est la science du raisonnement. Un paradigme logique est une nouvelle forme de programme, contenant des énoncés logiques.

La logique dont il est fait usage dans ces langages appartient au calcul des prédicats du premier ordre, soit une manière d'exprimer des énoncés logiques, c'est-à-dire vrais ou faux. A partir d'un ensemble d'énoncés, l'application de règles permet d'éta-

blir de nouveaux énoncés qui sont alors considérés comme prouvés. Ce mécanisme de déduction constitue la base pour trouver la solution d'un problème, en prouvant un énoncé qui matérialise cette solution. La solution d'un problème logique provient donc de l'ensemble des énoncés initiaux et ne dépend que de lui. La manière d'obtenir le résultat n'intervenant pas, le principe de base de la programmation logique est de limiter le programme aux seuls énoncés initiaux et de confier la dérivation de la solution à une tâche universelle qui dicte la manière de la conduire.

Un programme en langage logique consiste donc en un ensemble d'énoncés relatifs au résultat recherché. Sa particularité est qu'il n'indique rien sur la manière de trouver le résultat, c'est-à-dire sur la séquence particulière des instructions à exécuter afin d'y parvenir. La recherche du résultat proprement dit est confiée à un interpréteur, c'est-à-dire un programme qui analyse les énoncés et les résout [8]. La logique est assurée par le programme et le contrôle garanti par le système.

Les langages logiques représentent un pas vers la programmation véritablement automatique et prennent parfois le nom de «langages de très haut niveau». Certaines contraintes pratiques limitent cependant la programmation logique, notamment le manque d'universalité de la forme logique des énoncés et l'absence totale de contrôle, responsable d'une faible efficacité de calcul. Parmi les nombreux langages logiques développés, seul Prolog parvient à s'imposer.

SIMULA 67 (1965-1967)

La mise au point d'un langage passe généralement par la réutilisation ou la recomposition de concepts provenant d'un ou de plusieurs langages, complétés de diverses innovations. On retrouvera notamment dans Smalltalk les germes du paradigme orienté objet qui caractérisent ses glorieux prédécesseurs SIMULA et ALGOL.

SIMULA est conçu à Oslo par Dahl et Nygaard entre 1962 et 1967, afin de servir d'outil pour la simulation d'événements discrets. Son rôle sera par la suite étendu et il sera réécrit comme un langage de programmation universel. Bien que l'usage de SIMULA ne se soit jamais vraiment répandu, il aura toutefois eu une influence importante sur la méthodologie de la programmation, en introduisant notamment les concepts de la programmation orientée objet.

SIMULA offre en effet une généralisation de la notion de bloc, sous le nom de «classe». Comme le bloc ALGOL dont elle est inspirée, la classe Simula consiste en un ensemble de procédures et de déclarations de données suivi d'une séquence d'instructions exécutables encadrées par des parenthèses de début et de fin. Le bloc ALGOL et la classe SIMULA diffèrent cependant sur plusieurs points, notamment par les caractéristiques d'existence des données. Celle-ci est limitée à la durée de l'utilisation du bloc dans le premier cas, mais non pas dans le cas de la classe, ce qui constitue donc un avantage pour diverses applications, telles que la gestion de données ou l'exécution des tâches de systèmes de transport, de planification et de réservation. A partir de 1965, SIMULA est utilisé avec succès dans ces différents domaines.

Les travaux pour faire accéder SIMULA au rang de langage universel sont menés entre 1966 et 1967. C'est durant cette phase de développement que la classe SIMULA verra réellement le jour, en acquérant la propriété de pouvoir rassembler plusieurs sous-classes dans une superclasse (comme si l'on rassemblait les voitures, camions et vélos dans la classe des véhicules à roues). Forts de ces concepts et d'une bonne ténacité, Dahl et Nygaard parviennent en 1967 à développer un langage, SIMULA 67, qui sera formellement gelé l'année suivante. De nombreux ordinateurs de l'époque tels l'UNIVAC 1100, l'IBM 370, le CDC 6000 ou le DEC SYSTEM-10 seront notamment implémentés avec ce langage.

Smalltalk (1970)

Smalltalk est un développement effectué à Xerox PARC au Palo Alto Research Center. Les idées à la base de ce langage sont inspirées de SIMULA, mais proviennent aussi en grande partie de la vision de son créateur, Alan Kay [13]. Bien qu'il soit demeuré peu utilisé, Smalltalk a été déterminant dans deux domaines. Le premier est celui de la programmation orientée objet (OO); déjà introduits par SIMULA, les concepts OO sont exploités pleinement pour la première fois dans Smalltalk. L'autre domaine dans lequel ce langage a joué un rôle important est celui de l'environnement de programmation. Alan Kay entrevoit dès la fin des années 1960, un futur dans lequel les utilisateurs pourraient interagir avec des petites machines individuelles et puissantes. Introduisant l'usage intensif de fenêtres de dialogue, d'une souris et de menus, le système Smalltalk constitue le modèle de base à partir duquel se sont développés les systèmes de fenêtrage modernes. Steve Jobs, le patron d'Apple s'inspirera profondément de ce procédé lorsqu'il concevra les Macintosh destinés au grand public. Des Macintosh aux stations de travail X-windows, puis aux PC Windows, le fenêtrage se généralise progressivement, et s'impose au point d'être aujourd'hui une caractéristique évidente et obligatoire de tout ordinateur.

Paradigme des langages orientés objet (OO)

Après les paradigmes impératif, fonctionnel et logique, voici maintenant le paradigme orienté objet. La notion d'objet améliore la conceptualisation des tâches et permet d'aborder des opérations plus complexes.

Au cœur des langages OO prédomine la notion d'objet, définie comme une unité qui rassemble des places mémoires avec les opérations qui peuvent y accéder. Un objet simple est typiquement constitué d'une seule variable et des opérations destinées à en extraire sa valeur ou à lui en assigner une. La programmation OO est donc proche de la programmation impérative et s'oppose à la programmation fonctionnelle dans la mesure où elle met l'accent sur les places mémoire plutôt que sur les valeurs et les fonctions. La différence importante entre les langages impératifs et des langages OO réside dans le fait même de regrouper les variables et les opérations dans les

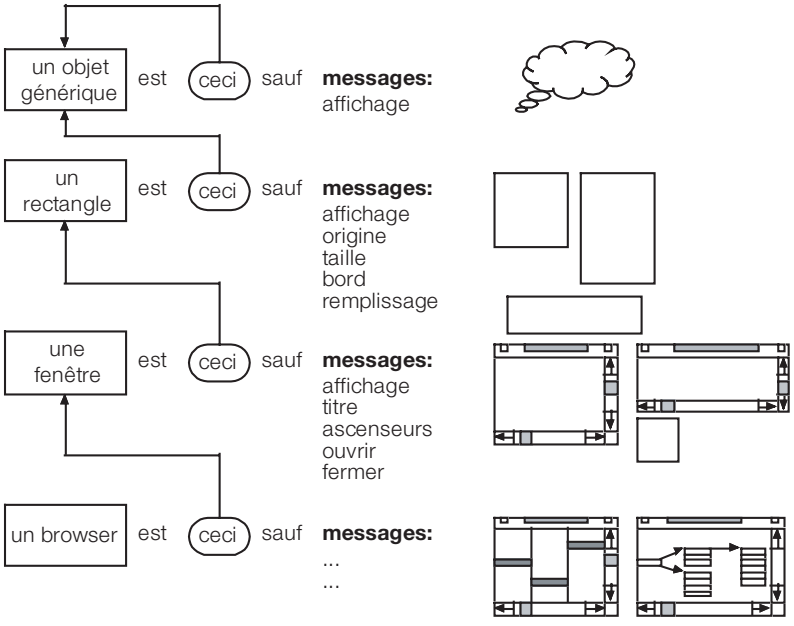


Fig. 5.20 Une classe d'objets hérite les propriétés d'une superclasse.

objets. Cette structure objet débouche sur une nouvelle forme de formulation et de perception d'une tâche de calcul. Elle peut en effet être exprimée comme une interaction entre les objets d'un même groupe, dont chacun se comporterait comme un ordinateur individuel, c'est-à-dire avec sa propre mémoire et ses propres opérateurs.

La plupart des langages orientés objet permettent de rassembler les objets aux propriétés communes dans des classes. La notion de classe simplifie la compréhension en favorisant la création de concepts avec une validité plus générale et augmente l'efficacité du système en mettant en commun certaines ressources. Deux mécanismes permettent notamment de différencier des classes. L'un est le mécanisme «d'héritage», qui permet de définir une classe à partir d'une autre par extension ou par restriction de certaines caractéristiques (fig. 5.20). L'autre est le mécanisme «polymorphisme», grâce auquel on peut définir une action qui a des réalisations différentes dans des objets différents.

Les caractéristiques de base du paradigme OO sont donc:

- notion d'objet,
- notion de classe.

Les avantages de la programmation OO se sont progressivement imposés et le paradigme des langages orientés objet est aujourd'hui le plus répandu en programmation. Ses représentants les plus caractéristiques sont Smalltalk, C++, Java.

C (1978)

C a été développé par Kernighan et Ritchie (K&R) à Bell Labs, le laboratoire de recherche de la compagnie Bell Telephone aux Etats-Unis. Développé à l'origine pour un ordinateur PDP 11 tournant UNIX, il est basé sur les langages cpl, bcpl et b, une famille de langages sans type. Le langage C n'est pas issu d'une longue réflexion théorique, mais a été conçu afin d'écrire des outils pour les ordinateurs au moyen d'un langage de haut niveau. Jusqu'alors, les systèmes d'exploitation étaient entièrement écrits en assembleur. Dès son apparition, C est employé à la réécriture du système d'exploitation UNIX. L'entreprise s'avère fructueuse puisque 95% de l'ensemble du système d'exploitation peut être pour la première fois formulé dans un langage de haut niveau. Ce succès fait la réputation de C, rapidement adopté par la communauté des programmeurs. C'est toujours le langage le plus connu au monde.

C voit le jour en 1978, mais un standard n'est défini que 11 ans plus tard. Ses contributions essentielles sont:

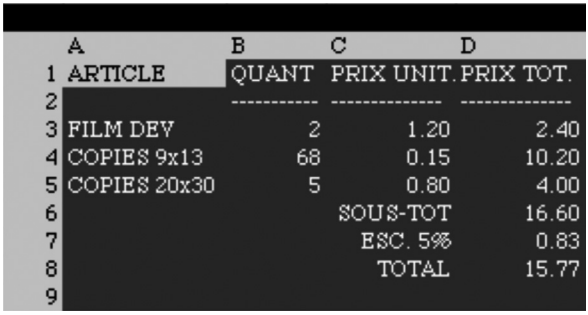
- l'efficacité,
- la conversion de type,
- les tableaux dynamiques,
- la bibliothèque,
- une bonne portabilité.

Ce langage offre cependant peu de facilité pour supporter la construction de grands ensembles de code; il aurait par ailleurs gagné à être un peu purifié, ce qui était techniquement possible à l'époque où il fut conçu.

Visicalc (1979)

Visicalc est le premier tableur électronique. Il a été inventé par le jeune Dan Bricklin, étudiant à l'Université de Harvard, dans le but de fournir un équivalent électronique aux tables comptables manuelles (fig. 5.21). Son succès est immédiat. Il devient très populaire et se répand conjointement aux ordinateurs personnels Apple qui viennent de faire leur apparition sur le marché. Un nouveau tableur, Multiplan, lui oppose toutefois une rude concurrence quelques mois plus tard. Ce dernier tourne sur les ordinateurs personnels IBM PC, et évoluera progressivement vers le tableur Excel.

L'arrivée du tableur est importante car elle marque la première familiarisation d'un large public avec une application non impérative. La plupart des langages en usage, comme BASIC, sont alors impératifs. Ils connaissent des valeurs en mémoire et des fonctions, et le programme dicte l'ordre des fonctions à calculer pour obtenir les valeurs finales. Les valeurs et fonctions de Visicalc ne diffèrent de celles de BASIC que dans la mesure où elles sont sont disposées dans un tableau visible sur l'écran de l'ordinateur. Mais la différence majeure entre application impérative et non impérative réside dans les règles d'évaluation des fonctions. Alors qu'avec BASIC, le pro-



	A	B	C	D
1	ARTICLE	QUANT	PRIX UNIT.	PRIX TOT.
2		---	---	---
3	FILM DEV	2	1.20	2.40
4	COPIES 9x13	68	0.15	10.20
5	COPIES 20x30	5	0.80	4.00
6			SOUS-TOT	16.60
7			ESC. 5%	0.83
8			TOTAL	15.77
9				

Fig. 5.21 Exemple Visicalc.

grammeur doit dicter l'ordre d'évaluation des fonctions, il en est complètement déchargé dans Visicalc. C'est l'implémentation qui doit veiller à trouver un ordre tel que toutes les fonctions trouvent des valeurs effectives à évaluer.

Les langages pour microprocesseurs

L'histoire de la programmation des ordinateurs se reproduit avec les microprocesseurs. On retrouve les étapes de programmation en code machine, en assembleur, puis en langage de haut niveau. Ce nouveau cycle évolutif se déroulera cependant beaucoup plus rapidement que le précédent.

Les microprocesseurs étant nés un peu par hasard, rien n'était prévu au départ pour les programmer. Les ingénieurs durent tout d'abord agir dans l'urgence, comme lorsque les premiers ordinateurs sont apparus. Programmés d'abord en code machine puis en assembleur, les microprocesseurs attendaient la venue d'un langage de haut niveau.

Lorsque Intel lance son premier microprocesseur en 1972, le 4004, Gary Kildall professeur en informatique à Monterey, en achète un et tente de le programmer. Il devient consultant à temps partiel chez Intel et écrit le premier langage de haut niveau pour le microprocesseur Intel-8008 en 1973. qu'il appelle PL/M (*Programming Language for Microcomputers*) un clin d'œil au langage PL/I d'IBM.

1975 voit l'arrivée d'une implémentation de BASIC pour microprocesseurs Intel 8080 et Z-80, réalisée par Li Chen Wang. Sa dénomination «Tiny BASIC» est liée à son très faible encombrement: le programme prend place dans 2k octets de mémoire seulement, offrant encore autant de mémoire libre pour les données dans une configuration typique d'un microprocesseur de l'époque. Tiny BASIC est en outre le premier logiciel *freeware* connu.

Un jeune étudiant à Harward et un employé de Honeywell, Bill Gates et Paul Allen, réalisent eux aussi le potentiel de BASIC. Ils apprennent l'existence de l'Altair 8800 lorsque celui-ci est présenté à la une du journal *Popular Electronics* en janvier

1975, en tant que premier mini-ordinateur vendu en kit. Les deux passionnés d'informatique imaginent alors que cet ordinateur a besoin de logiciel. Ils contactent le directeur du constructeur MITS, Ed Roberts, pour lui proposer un BASIC qu'il pourrait faire tourner sur l'Altair. Roberts se déclare prêt à acheter ce langage s'il s'avère opérationnel. Gates et Allen se lancent dans le projet et parviennent à le mener à bien. Grâce aux revenus des licences vendues à MITS, ils fondent Microsoft Co. et commencent à produire BASIC pour d'autres plates-formes. A la fin des années 70, BASIC aura été porté sur les ordinateurs Apple, Commodore et Atari. Le règne de Microsoft commence.

BASIC est l'un des premiers produits vendus par Microsoft, mais aussi le symbole du premier cas important de piraterie logicielle. Après que Bill Gates en eut égaré une copie sur bande perforée lors d'une exposition, ce programme fit l'objet d'une duplication importante avant même sa mise sur le marché.

Ada (1980-1987)

La genèse des langages peut être très diverse. ADA est le résultat d'un gigantesque effort de planification, alors que C++ et Java proviennent d'initiatives individuelles.

ADA est le fruit des efforts du Département de la défense américaine pour produire un langage puissant et expressif basé sur des standards bien établis. Soutenu par l'énorme budget de la défense américaine, ce langage repose sur une spécification très poussée. Le premier rapport relatif à Ada date de 1980; sa spécification ISO 8652 aura lieu 7 ans plus tard. Les standards ISO de 1995 concernent des changements dans les domaines suivants:

- support de jeux de caractères 8 et 16 bits,
- programmation orientée objet avec polymorphisme temps réel,
- extension de types d'accès,
- synchronisation orientée sur les données,
- interfaçage vers d'autres langages.

Le nom Ada vient de Lady Augusta Ada Byron, comtesse de Lovelace et fille du poète anglais Byron, qui parvint à programmer la machine analytique conçue par Charles Babbage.

C++ (1986)

C++ est le résultat des efforts de Bjarne Stroustrup chez AT&T Bell Labs pour produire une version de C orientée objet. Celui-ci avait été exposé précédemment au SIMULA de la fin des années 1960 et réalisait les avantages d'un langage de ce type. Les caractéristiques de C++ à l'origine de son succès sont les suivantes:

- extension largement compatible avec ANSI C,

- support des concepts orientés objet,
- langage fortement typé,
- une bonne disponibilité des compilateurs.

Un standard de C++ n'est disponible que depuis novembre 1997.

Java (1994)

Vers la fin des années 1980, James Gosling, créateur de Emacs, un éditeur de texte fort connu des programmeurs, est frustré par l'implémentation en C++ d'un nouvel éditeur. Il crée un nouveau langage nommé Oak. A la même époque chez Sun, Bill Joy décide que C++ est trop compliqué et se met à la recherche d'un nouvel environnement de développement. Sun adopte alors Oak pour certains projets et le développe durant quelques années. Rebaptisé Java, il est utilisé par Sun lors de la conception du browser Hotjava. Les deux produits sont lancés conjointement en 1994, et Java s'impose dès lors comme un langage à part entière.

Java est un langage orienté objet relativement simple (fig. 5.22). Issu de C++, il est délivré de la plupart des caractéristiques dangereuses de ce dernier, il est profondément orienté objet et ne comporte que des classes. Le langage est interprété et le code binaire intermédiaire est prévu pour tourner sur toute machine virtuelle adéquate. Java propose donc une portabilité au niveau du code objet et diffère en cela des autres langages qui offrent généralement une compatibilité au niveau du code source seulement. Citons quelques-unes de ses caractéristiques:

- un garbage collection automatique, pas de libération de mémoire nécessaire,
- aucun pointeur, seulement des références,
- un langage multitraitement, avantage au niveau de l'application,
- l'inclusion d'un environnement à fenêtres nommé AWT; en ceci comparable à VisualBASIC mais dans le cadre d'un langage plus puissant.

Java est le langage de l'Internet. Ses inconvénients sont ceux de tout nouveau langage, notamment ceux de son développement drastique et du grand nombre de versions différentes qui en ont résulté. Certaines machines virtuelles ne sont par ailleurs pas compatibles avec Java.

Les concepts

Lors de l'apparition des ordinateurs, tout reste à inventer en matière de langage de programmation. Konrad Zuse fait figure de précurseur en développant le langage Plankalkül, qui demeurera malheureusement dans l'ombre. FORTRAN, créé chez IBM sous l'impulsion de John Backus, est le premier langage à être largement diffusé et à prendre une signification économique considérable. De nombreuses tentatives sont ensuite entreprises afin de remédier aux insuffisances de ce langage. ALGOL,

```
class sum {
        static int result=0;
        static int tab1[]= new int[100];
        public static void main(String args[]) {
                tab1[0]=23;
                tab1[1]=34;
                tab1[2]=9;
                tab1[3]=7;
                sumOdds();
                System.out.println(result);
        }
        static int sumOdds () {
                for (int i=0; i<100; i++){
                        if ((tab1[i]%2)!=0){
                                result=result+tab1[i];
                        }
                }
                return result;
        }
}
```

Fig. 5.22 Exemple Java.

COBOL, PASCAL et BASIC offrent des améliorations tout en poursuivant le même paradigme impératif. La recherche d'une nouvelle approche s'attachant à la formulation d'un problème sans en détailler toutes les étapes conduit aux paradigmes fonctionnel et logique. LISP est un précurseur toujours actuel des langages fonctionnels et Prolog celui des langages logiques. La confrontation de ces programmes avec des problèmes complexes a cependant mis en évidence les insuffisances des uns et des autres. Une approche modulaire liée à des objets introduite par SIMULA et complétée ensuite dans Smalltalk démontre l'intérêt de l'approche orientée objet et lance les développements qui conduiront à Ada, C++ et Java. Ces langages jouent aujourd'hui un rôle prépondérant, mais n'empêchent nullement la coexistence avec de nombreux autres langages. Les différentes approches offertes par ces langages ne doivent pas faire oublier que la qualité du programme dépend surtout de celle du programmeur [6].

Les systèmes d'exploitation (OS)

Une fois les programmes disponibles, il reste à organiser leur exécution rationnelle par une exploitation efficace des ordinateurs. Organisée à l'origine par l'intermédiaire d'une simple feuille de réservation, cette exploitation constitue peu à peu une tâche à part entière, réglée par des programmes dont l'ensemble est appelé système d'exploitation (ou OS, pour Operating System) [14][15][17].

La feuille de réservation

Lorsqu'il voit le jour, l'ordinateur est disponible sur réservation et pour une seule tâche à la fois.

Faire tourner un programme sur un des premiers ordinateurs constitue un travail laborieux comprenant de multiples étapes: encoder le programme sur des cartes perforées, charger le code en mémoire au moyen d'un lecteur de cartes, choisir manuellement l'adresse de départ, lancer l'ordinateur, vérifier son fonctionnement sur la console, éventuellement intervenir en cas d'erreur, et, à l'arrêt du programme, recueillir les résultats. C'est généralement le programmeur lui-même qui opère la machine, de manière presque interactive. Le partage de la machine entre les programmeurs est réglé sur la base d'une simple feuille de papier sur laquelle chacun vient inscrire ses heures de réservation (fig. 5.23).

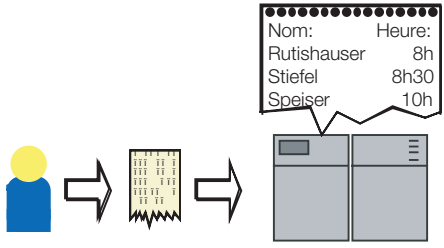


Fig. 5.23 Une simple feuille de réservation pour tout OS.

Les seules aides alors disponibles sont des routines relatives à l'entrée et la sortie des données, telles que des programmes écrivant en mémoire les codes lus sur un lecteur de cartes. Le système d'exploitation se limite ainsi à quelques centaines d'instructions machine. Cette forme d'OS primitif n'en est donc pas encore vraiment un. Le principal problème qu'il pose est la mauvaise utilisation des ressources du processeur. Celui-ci est presque toujours "en attente", car la mise en place du calcul dépasse de loin le temps de calcul effectif. Ce mode d'exploitation demeurera pourtant en vigueur jusqu'au milieu des années 1950.

Les OS batch

Avec les systèmes d'exploitation batch, l'ordinateur reçoit plusieurs tâches, disposées en queue et traitées en séquence.

Les programmes ou tâches à effectuer successivement par l'ordinateur sont mis bout à bout et forment un lot (*batch* en anglais) destiné à être traité sans intervention humaine. La figure 5.24 illustre un batch de deux tâches matérialisé par des cartes perforées mises bout à bout.

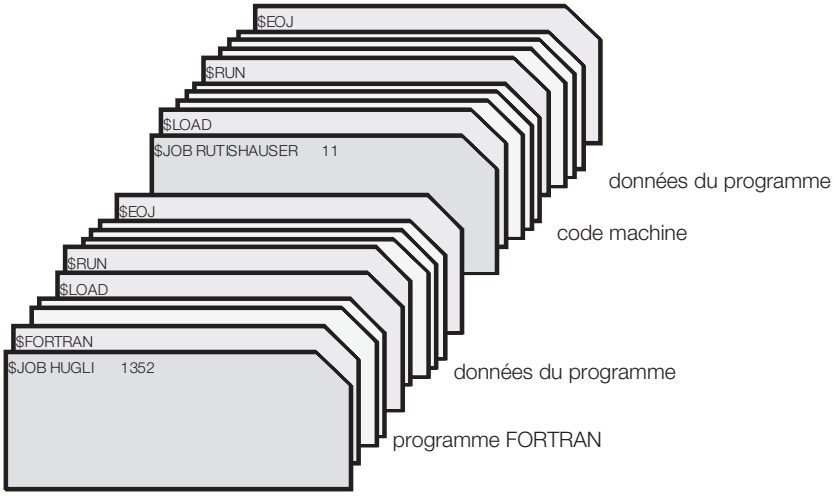


Fig. 5.24 Cartes perforées constituant un batch de deux jobs.

Le fonctionnement de l'OS batch est assuré par un programme placé en mémoire et appelé moniteur. Il constitue une première ébauche de système d'exploitation et a pour tâche de lire et interpréter une à une les instructions des tâches batch, notamment le chargement des données et codes en mémoire de l'ordinateur et le contrôle du programme nouvellement chargé.

Ce fonctionnement requiert une double occupation de la mémoire: une première partie est utilisée en permanence par le moniteur, alors que la deuxième est employée de manière variable pour l'exécution successive de chaque programme (fig. 5.25).

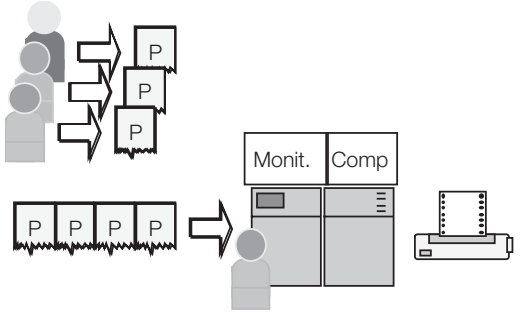


Fig. 5.25 OS batch.

Le premier OS batch est développé au milieu des années 1950 par General Motors pour être utilisé sur un IBM 701. D'autres clients d'IBM l'améliorent et le portent sur

IBM 704. Ce n'est qu'au début des années 1960 que les vendeurs d'ordinateurs développent des systèmes d'exploitation pour leurs ordinateurs. IBM renomme alors IBSYS le moniteur développé par ses clients et le destine aux ordinateurs 7090/7094.

Au fil du temps, les OS bénéficient de deux améliorations. La possibilité d'écrire les lots de tâches directement sur bande magnétique permet d'accélérer le fonctionnement. L'interruption d'exécution à partir d'une horloge permet quant à elle de sortir automatiquement de programmes ne terminant pas dans les temps. L'inconvénient majeur de ces systèmes reste cependant la lenteur des lecteurs qui limite la pleine exploitation des ressources du processeur.

Les OS batch avec multiprogrammation (1960-1980)

L'ordinateur parvient bientôt à traiter plusieurs tâches à la fois.

Au fur et à mesure de l'augmentation de la mémoire disponible pour la machine, il devient possible de généraliser l'idée d'un accès à plusieurs travaux chargés simultanément dans la mémoire. Le résultat est la «multiprogrammation». Le processeur passe d'une tâche à une autre suivant un certain ordre et en mettant à disposition de chacune d'elles les ressources de calcul pendant un temps bien défini (fig. 5.26).

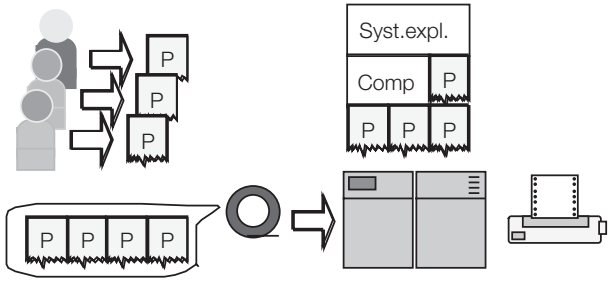


Fig. 5.26 OS batch avec multiprogrammation.

A ce stade de développement, le moniteur a augmenté de taille, et ressemble à un système d'exploitation moderne. Il est notamment capable:

- de démarrer les tâches de chaque utilisateur,
- d'effectuer des opérations de spooling,
- de prendre en charge des fonctions d'entrée et de sortie pour le compte des utilisateurs,
- de commuter les différentes tâches,
- d'assurer la protection des données en exerçant ce qui précède.

Les OS à temps partagé (1965-aujourd'hui)

Avec les OS à temps partagé, l'ordinateur communique directement avec l'utilisateur. Les tâches sont toutes traitées à la fois et chaque utilisateur peut suivre le déroulement de son propres travail.

Avec les OS batch, l'utilisateur n'a aucune influence directe sur le déroulement du programme alors que de nombreuses tâches nécessitent qu'il puisse intervenir durant leur déroulement au travers d'un terminal. C'est aujourd'hui possible avec un ordinateur personnel à la disposition de chaque utilisateur. Dans les années 1960, les ordinateurs sont gros et chers. On cherche donc à partager le temps de l'ordinateur entre les utilisateurs: c'est l'avènement du fonctionnement à temps partagé (*time-sharing*).

Puisque la multiprogrammation permet de traiter plusieurs tâches simultanément, elle peut être aussi utilisée pour traiter les tâches de plusieurs utilisateurs. Il suffit dans ce cas d'attribuer à chaque utilisateur une tâche spécifique. L'interactivité demandée aux systèmes à temps partagé (notamment parce que l'utilisateur attend une réponse rapide à ses actions) impose une contrainte temporelle nouvelle au séquencement des programmes.

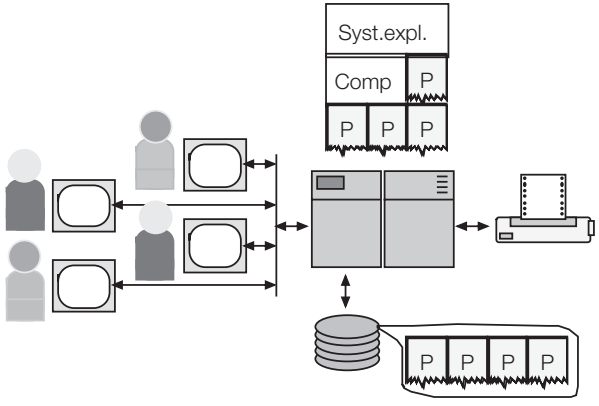


Fig. 5.27 Système à temps partagé.

CTSS (*Compatible Time Sharing System*) est l'un des premiers systèmes à temps partagé (fig. 5.27). Conçu par une équipe du MIT dirigée par Fernando Corbato, il est développé dans le cadre du projet MAC (*Machine Aided Cognition*) en 1961-1963. Prévu d'abord pour l'IBM 709, il est porté ensuite sur l'IBM 7094. En comparaison des systèmes ultérieurs, cet OS est relativement simple et petit. Prévu pour tourner sur 32 k de mémoire (36 bits), le moniteur résident ne fait que 5k. Une interruption se produit toutes les 200 ms. Le programme associé à chaque utilisateur est alors chargé successivement en mémoire à partir du disque.

Le développement et l'évolution des OS

Produits au carrefour du possible et du souhaitable, les OS connaissent des fortunes fort diverses. Les deux raisons principales de l'échec de certains d'entre eux sont le gigantisme et l'obsolescence.

La figure 5.28 illustre l'évolution de quelques systèmes d'exploitation importants, et met en avant l'évolution de l'une de leur caractéristiques fondamentales: la taille. Issus d'un compromis entre les innombrables fonctionnalités souhaitables et les possibilités limitées de leur réalisation, les systèmes d'exploitation ont du succès en rapport direct avec l'adéquation du compromis trouvé.

Les premiers systèmes d'exploitation sont des systèmes batch développés dans des universités: EDSAC à l'Université de Cambridge et Atlas à l'Université de Manchester. CTSS est l'un des premiers systèmes à temps partagé. SCOPE, système batch commercialisé par la firme Control Data Corporation, est quant à lui un des premiers systèmes d'exploitation commerciaux.

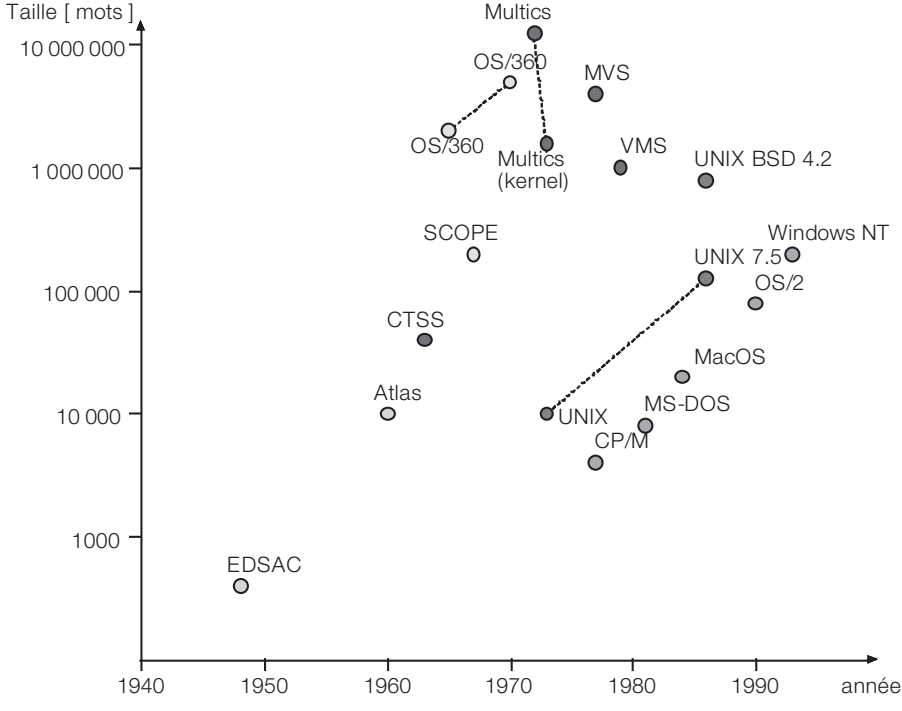


Fig. 5.28 Evolution des systèmes d'exploitation.

Le système MULTICS, développé en 1973 conjointement par le MIT et Bell Labs, illustre bien la course en avant vers des fonctionnalités toujours plus grandes. Ses concepteurs voient grand et envisagent des fonctions très modernes, notamment liées à la sécurité et l'interfaçage avec l'utilisateur. Hélas, le système est trop en avance. MULTICS atteint la taille énorme, à l'époque, de 20 millions d'instructions et il est abandonné par les concepteurs. Commercialisé quelque temps par Honeywell, c'est en effet un échec commercial.

Des systèmes moins ambitieux comme OS/360 et MVS ont plus de chance, et montrent qu'un bon compromis est trouvé pour les gros ordinateurs. Avec l'arrivée des mini-ordinateurs, le développement des systèmes d'exploitation propose des OS type UNIX et VMS. CP/M et MS-DOS seront les systèmes adoptés pour les micro-ordinateurs.

OS/360 – MVS

«I think there is a world market for maybe five computers.»
Thomas Watson, directeur d'IBM, 1943

IBM a une bonne situation sur le marché des ordinateurs avec sa série 7000 quand, en 1964, le constructeur lance le nouveau système 360. Pour certains clients, c'est une mauvaise nouvelle, car il est incompatible avec la série précédente. Ce pas est osé, mais IBM le juge nécessaire pour se libérer de certaines contraintes et mettre sur pied un système capable d'évoluer avec la technologie des circuits intégrés.

Le système d'exploitation prévu pour cette nouvelle génération d'ordinateurs est OS/360 [14]. Son développement se révèle toutefois très difficile. Commencé en 1964, il ne s'achève qu'en 1967. C'est d'abord et pour quelque temps un système batch avec multiprogrammation avant de devenir à temps partagé. Il comprend plus d'un million d'instructions machine. Sa forme évolue et aboutit à la version la plus complète, lancée en 1969, nommée MVT (*Multiprogramming with a Variable number of Tasks*). Avec ce système, l'allocation mémoire des tâches est variable et ne doit pas être décidée jusqu'à l'exécution. C'est le système d'exploitation par excellence des gros 360 et des premiers 370. MVT n'est pas le plus évolué des systèmes d'exploitation contemporains, mais il se caractérise par de grandes qualités opérationnelles.

Une autre évolution de OS/360 introduit la mémoire virtuelle, une technique consistant à mettre à la disposition des programmes une mémoire logique plus grande que la mémoire physique (fig. 5.29). Le programme, stocké sur le disque, est divisé en pages comme un livre. Seules les pages en cours d'utilisation sont stockées en mémoire. Un mécanisme automatique de pagination traduit une adresse virtuelle dans l'adresse physique de la mémoire correspondante et prend en charge la gestion des pages [15].

C'est MVS qui consacre la mémoire virtuelle en accordant un domaine d'adressage virtuel de 16 MBytes à chaque tâche. La limite est encore liée à l'adresse de 24 bits de l'architecture du processeur 370. Une extension majeure de cette limite arrive

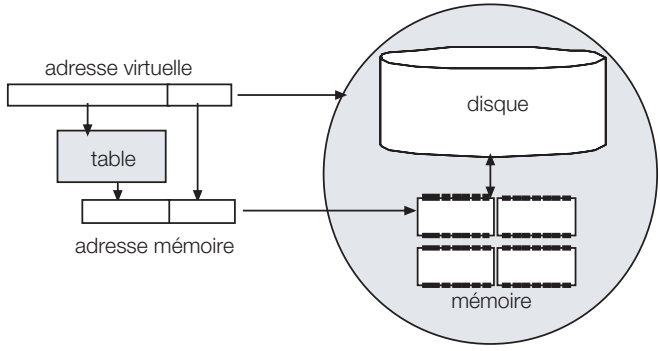


Fig. 5.29 Principe de la mémoire virtuelle.

finalement en 1983. Un MVS étendu désormais nommé MVS/XA considère dorénavant une adresse virtuelle de 31 bits, avec pour conséquence une capacité d'adressage virtuel par tâche de 2000 MBytes. MVS est probablement le système d'exploitation le plus grand et le plus complexe développé à cette époque.

UNIX

Quand la vague des mini-ordinateurs succède à celle des ordinateurs, les OS doivent s'adapter à ces nouvelles machines aux ressources plus limitées. UNIX et VMS sont développés spécialement pour ce nouveau marché.

UNIX est développé en 1969 chez Bell Labs sur mini-ordinateur DEC PDP-7. Son nom est choisi en opposition à MULTICS, le système du MIT. L'équipe de Bell Labs comprend notamment à cette époque d'anciens collaborateurs du projet MAC du MIT qui ont abandonné MULTICS. Bien qu'UNIX soit considéré comme un petit MULTICS, ses développeurs avouent s'être inspirés plus encore de CTSS.

La première étape importante est le portage réussi de l'OS du mini-ordinateur PDP-7 au PDP-11, qui laisse présager de la possibilité à court terme de porter UNIX sur différents types d'ordinateurs. L'étape suivante fut la réécriture d'UNIX en langage C. C'est la première fois qu'un logiciel aussi critique qu'un système d'exploitation est transcrit dans un langage de haut niveau. Cet OS suscite rapidement un large intérêt. Une première description d'UNIX apparaît dans une revue technique en 1974. Des licences sont vendues à certaines entreprises et mises à disposition de quelques universités. Le développement d'UNIX est poursuivi par AT&T, dont la version 7 est distribuée dès 1978. C'est cependant à partir de ce moment que la maîtrise d'UNIX échappera définitivement à AT&T, les nombreux développements menés de front par d'autres acteurs conduisant à la dispersion de variantes incompatibles. Un développement important est notamment mené à l'Université de Californie à Berkeley et donne

naissance à la version UNIX BSD, sensiblement différente de la version UNIX System V de AT&T. UNIX étant une marque déposée par AT&T, de nombreux constructeurs se lancent parallèlement dans la commercialisation de leur propre version du logiciel. Des problèmes d'incompatibilité ne tardent pas à apparaître. Afin d'y remédier, un projet d'uniformisation est mis en place, et débouchera dès 1990 aux standards POSIX de l'organisation OSF (Open Software Foundation). La facilité de portage de code entre les différents types d'UNIX offerte par ces normes participeront dès lors à la large utilisation de ce système d'exploitation.

VMS

«There is no reason anyone would want a computer in their home.»
Ken Olson, PDG et fondateur de Digital Equipment Corp., 1977

VMS (*Virtual Memory System*) est le système d'exploitation de DEC (Digital Equipment Corp) créé pour sa famille d'ordinateurs VAX à la fin des années 1970. Il se distingue des systèmes d'exploitation précédemment en vigueur pour les PDP, notamment par le support de mémoire virtuelle utilisé (fig. 5.30).

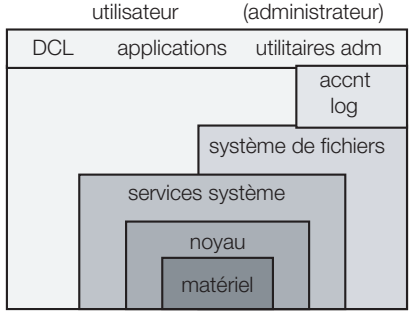


Fig. 5.30 Structure en couches de VMS, typique d'un OS.

Créé pour le modèle VAX-11/780, ce système d'exploitation est resté en vigueur pour toute la famille des ordinateurs DEC, de la micro-VAX aux machines de la série 9000, systèmes multiprocesseurs ou en grappe inclus.

CP/M

Quand arrivent les micro-ordinateurs, un nouvel OS doit être développé pour ces machines. CP/M est le premier sur le marché, avant de se faire dépasser par MS-DOS.

Gary Kildall, l'inventeur du premier langage de haut niveau pour microprocesseurs, poursuit ses développements sur un système à base d'Intel-8080 comprenant notamment un disque souple de Shugart. Au moyen de son langage PL/M, il écrit le

premier système d'exploitation pour microprocesseurs, qu'il termine en 1974. Il le nomme CP/M et l'offre à Intel pour 20 dollars. Intel décline cette proposition. Deux ans plus tard, Kildall et sa femme fondent Digital Research pour le commercialiser à leur propre compte. Le produit rencontre un vif succès et est vendu à de nombreux fabricants. Il est complété avec un BIOS (Basic Input Output System), un éditeur, un assembleur, un débuggeur et des utilitaires. Système d'exploitation des microprocesseurs 8-bits par excellence, CP/M a tourné sur plus de 3000 machines différentes et a ainsi offert une plate-forme commune pour des machines par ailleurs incompatibles, favorisant ainsi la dispersion de nombreuses applications. Avec l'arrivée du processeur Intel-8086, le système d'exploitation est étendu à l'architecture 16 bits et devient CP/M-86. C'est à la même époque que IBM souhaite commercialiser son ordinateur personnel IBM-PC et cherche un système d'exploitation pour celui-ci. Invité, le président de Digital Research décline l'offre et c'est le président de la petite compagnie Microsoft, Bill Gates qui est reçu par le géant de l'informatique.

MS-DOS

«640K (of memory) ought to be enough for anybody.»
Bill Gates, PDG et fondateur de Microsoft Co., 1981

Bill Gates relève le défi, bien que n'ayant jamais encore écrit de système d'exploitation. Microsoft contacte Seattle Computer, une entreprise qui possède un système d'exploitation pour les processeurs 16 bits, compatible avec CP/M: QDOS. Microsoft achète les droits de QDOS, engage ses créateurs, et modifie ce système de manière à couvrir les besoins d'IBM. Le système d'exploitation DOS (Disc OS) est né. Il prend le nom de PC/DOS pour les ordinateurs d'IBM, mais sera commercialisé sous le nom de MS-DOS pour d'autres PC.

MS-DOS a évolué au cours de ses différentes versions. La version 1, distribuée dès août 1981, ressemble beaucoup à CP/M. Elle consiste en 4000 lignes de code assembleur, tourne avec 8 k de mémoire et supporte les disquettes floppy. Le catalogue est encore limité à 64 fichiers (fig. 5.31).

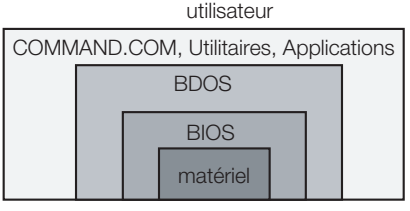


Fig. 5.31 Système d'exploitation de type DOS.

Avec le développement du PC-XT chez IBM, basé sur un disque dur, Microsoft lance une version 2 qui inclut certains concepts d'UNIX. Les fichiers peuvent être

organisés en répertoires hiérarchiques et la partie résidente en mémoire passe à 24 k octets. La version 3 de MS-DOS arrive en 1984, conjointement avec le lancement du PC-AT d'IBM, basé sur le processeur Intel 80386. Outre l'ajout de quelques fonctionnalités comme le support de réseau, le système d'exploitation ne tire pas profit des avantages de l'architecture des nouveaux processeurs. DOS 3.3 lancé en 1987 et destiné au support des nouvelles machines d'IBM, les PS/2, ne tiendra pas plus compte des possibilités nouvelles offertes par les processeurs 32 bits supportés.

La version originale de MS-DOS tournait sur les processeurs Intel 8086/8088. Ces circuits, construits selon une architecture à 16 bits, n'ont qu'une adresse limitée à 20 bits, soit une mémoire adressable de 1 million d'octets au maximum. L'évolution de MS-DOS vers un produit plus moderne est profondément marquée par cette limitation originale de l'espace d'adressage.

Evolution des OS

Peu à peu, les OS intègrent de nouvelles fonctionnalités et supportent les ressources de calcul de plusieurs processeurs et ordinateurs. Un OS temps réel offre des délais de réponse maximaux garantis à des événements critiques et s'ouvre ainsi aux exigences des environnements de contrôle et commande. Un OS multitâche met la multiprogrammation au service du programmeur et lui offre une grande flexibilité de partage du travail entre plusieurs tâches exécutées conjointement. Un OS multiprocesseur rend disponibles ses services en s'appuyant sur les ressources de plusieurs processeurs et augmente ainsi les performances globales de calcul. Un OS distribué gère en commun et tire parti des ressources de plusieurs ordinateurs mis en réseau.

Au cours des années 1990, les différences entre gros ordinateurs, mini-ordinateurs et ordinateurs personnels s'estompent peu à peu. Les grands constructeurs adoptent tous UNIX, et les OS s'unifient. Les produits vendus aux clients sont des implémentations propriétaires d'UNIX très différentes entre elles, mais dont la portabilité est assurée au travers de standards, notamment POSIX. On trouve ainsi AIX chez IBM, Ultrix chez DEC, HP-UX chez HP, IRIX chez Silicon Graphics et SUN OS (puis Solaris) chez Sun. Quant aux OS des ordinateurs personnels, ils comptent parmi eux le MacOS d'Apple ainsi que les successeurs Microsoft de MS-DOS qui s'appellent successivement Windows 3, OS/2 (issu d'un développement commun avec IBM vite abandonné) et Windows avec les appellations 95, 98, NT, 2000 et XP.

On doit à Linus Torvalds un récent et étonnant développement: la création d'un OS libre et gratuit accessible à tous. Il débute sa conception à partir du code source de MINIX, puis le diffuse sur l'Internet. Il récolte par ce biais des contributions de développeurs du monde entier. Le résultat s'appelle Linux et se présente en 2002 sous une douzaine de versions et plusieurs formes d'interfaces à choix, en combinant les avantages d'UNIX et de Windows. Le succès de cet OS est aujourd'hui sans cesse grandissant.

Les concepts

Rien ne permettait de prévoir les OS avant l'arrivée des ordinateurs. Il fallut encore attendre près de dix ans après leur apparition pour que les premiers systèmes d'exploitation dignes de ce nom soient opérationnels. Ces systèmes ne sont pas nés chez les constructeurs d'ordinateurs, mais plutôt chez les clients qui avaient le souci d'exploiter ces machines.

Les développeurs d'OS sont rapidement confrontés à des questions de complexité limitant la réalisation des fonctionnalités souhaitables dans le système. L'histoire des systèmes d'exploitation est marquée par la recherche du souhaitable réalisable. Orientée nécessairement vers les gros ordinateurs au début, cette quête s'est répétée avec les mini-ordinateurs puis avec les ordinateurs individuels.

Les OS sont parmi les ensembles de logiciels les plus complexes qui soient. Ils constituent aujourd'hui encore l'un des plus grands défis de l'informatique et sont l'objet d'âpres batailles commerciales.

Références

[1] Jerome A. Feldman, «Programming Languages», *Scientific American*, Vol. 241, N° 6, Dec. 1979.

[2] Michel Gauthier, «Le saut explicite en programmation: essai de synthèse», TSI, Vol. 5, n° 5, 1986.

[3] Robert L. Glass 1998. In The Beginning: Recollections of Software Pioneers. Los Alamitos, California: IEEE Computer Society.

[4] David Harel, *Algorithimcs: The spirit of Computing*, Addison-Wesley, 1987.

[5] History of Programming Languages Conference, ACM SIGPLAN Notices, Vol. 13, N° 8, 1978

[6] C.A.R. Hoare, «Programming: Sorcery or Science», IEEE Software, April 1984, pp. 5-16.

[7] James Hunt, «Programming Languages», *Computer*, April 1982, pp. 70-88.

[8] K. C. Louden, *Programming languages, Principles and Practice*, PWS-KENT Publishing Company, Boston, 1993.

[9] Peter Wegner, *Programming Languages – The First 25 Years*, IEEE Trans. on Computers, Vol C-25, N° 12, Déc. 1976.

[10] Philippe Breton, *Une histoire de l'informatique*, Editions du Seuil, Coll. Sciences Points, 1990.

[11] R. L. Wexelblat, *History of programming languages*, Academic Press, 1981.

[12] Jean E. Sammet, *Programming Languages: History and Fundamentals*, Englewood Cliffs, N. J.: Prentice-Hall. 1969.

[13] «Special Issue: Computer Software», *Scientific American*, September 1984, Vol. 251, pp. 40-160.

[14] Silberschatz & Galvin, *Operating System Concepts*, Addison-Wesley, 1998.

[15] William Stallings, *Operating Systems*, Prentice Hall, 1995.

[16] T. J. Bergin & R. G. Gibson, *History of programming languages*, Tome I & II, Addison-Wesley Publishing Company, 1996.

[17] Tanenbaum & Woodhull, *Operating Systems: Design and Implementation*, Prentice-Hall, 1997.

[18] «Structured Programming», *Computer*, June 1975, Volume 8, N° 6, pp. 1-73.

[19] IEEE Annals of the History of Computing, Journal of the IEEE Computer Society, 1979-2000.

[20] Coding History: http://www.heuse.com/coding.htm

[21] Manchester Baby Computer: http://www.computer50.org/

CHAPITRE 6

HISTOIRE DES MICROPROCESSEURS

«Annoncer une nouvelle ère de l'électronique, rétrospectivement, ce n'était pas si faux.»
Ted Hoff, à propos de l'annonce du 4004, le premier microprocesseur.

L'invention du transistor et de la microélectronique

L'apparition du transistor en 1947, suivie de celle du circuit intégré en 1958, rendent alors possible l'invention du microprocesseur. Il ne constitue pas une nouvelle architecture de machine de traitement de l'information, mais son ingéniosité réside dans le fait de concentrer sur une seule puce de silicium l'ensemble des fonctions d'un ordinateur, soient les unités de commande et de traitement. Les architectures des premiers microprocesseurs sont ainsi héritées des premiers ordinateurs, notamment ceux de Von Neumann et de Harvard. Le principe de ces architectures reste basé sur l'addition de deux nombres, mais l'opération est désormais effectuée sur une puce de silicium, appelée circuit intégré, plutôt que d'être réalisée dans une immense armoire, comme celle de l'ordinateur «Baby», comportant des milliers d'éléments et de tubes à vides.

Invention du transistor

William Shockley, John Bardeen et Walter H. Brattein reçoivent le prix Nobel en 1956 pour leur invention du 16 décembre 1947, le transistor (pour TRANSfer resISTOR, résistance de transfert) [15].

L'invention est présentée au management de leur compagnie Bell Telephone Labs le 23 décembre de la même année, autre date parfois retenue pour l'invention du transistor. La compagnie n'annonce la nouvelle que le 20 juin 1948, dans quatre courts paragraphes à l'avant-dernière page du *New York Times*. Les journaux techniques sont eux aussi très lents à divulguer la nouvelle. Ce manque d'enthousiasme pousse les Bell Labs à offrir gratuitement une licence d'exploitation du transistor à ceux désirant l'utiliser, et à multiplier les séminaires pour faire connaître ce nouveau dispositif. Mais le tube à vide n'en demeure pas moins perçu comme suffisant, robuste et bien connu, et il oppose une sévère concurrence au transistor, dont bien peu à l'époque imaginent l'avenir auquel il est promis.

Le transistor inventé aux Bell Labs est «à pointes», formé d'une pastille de germanium sur laquelle sont déposées deux pointes métalliques très rapprochées. Il est impossible à produire industriellement. Shockley, quelque peu vexé de n'avoir participé que de loin à son invention parce que travaillant sur une autre structure MOS,

décide d'apporter sa contribution. Après un travail acharné, il invente en janvier 1948 le transistor bipolaire, plus simple à fabriquer. Il est proposé par les Bells Labs à partir de 1951 et sera employé dans tous les circuits intégrés jusqu'au début des années 1970, avant d'être remplacé par les transistors MOS.

Au début des années 1950, le transistor est toujours le mal aimé des ingénieurs. Le premier circuit à en être doté est une aide auditive, mise au point par la compagnie Sonotone en février 1953, comportant 5 transistors et deux tubes à vide. Il faut attendre le milieu des années 1950 pour que des compagnies telles que General Electric, Sylvana et RCA produisent des transistors en germanium. Texas Instruments met sur le marché le premier transistor en silicium en 1954. Encouragé par le développement du marché des transistors, Shockley fonde cette même année sa propre compagnie et s'installe à Palo Alto, en Californie, marquant ainsi le début de la Silicon Valley. Shockley engage de jeunes chercheurs comme Moore et Noyce, mais l'association ne dure pas, ces derniers lui reprochant vite son manque de vision quant aux perspectives commerciales offertes par les transistors. Les «traîtres», comme les appelle Shockley, partent fonder leur propre compagnie, Fairchild, en 1957. Ils inventent en 1959 un nouveau procédé très avantageux pour la fabrication de transistors, appelé «planaire». Plus tard, Moore et Noyce fondent la compagnie Intel, qui fabrique aujourd'hui les microprocesseurs Pentium.

Les transistors des années 1950 étaient fabriqués isolément. En 1958, Kilby (Prix Nobel de physique 2000), de Texas Instruments, réalise le premier circuit intégré doté de plusieurs transistors sur la même puce. Il est suivi en 1960 par Noyce, qui réalise des circuits intégrés commerciaux.

Description d'un transistor

Notre description se bornera au transistor MOS (Metal Oxide Semiconductor), élément utilisé aujourd'hui pour les microprocesseurs. Celui-ci permet un contrôle du courant électrique passant entre deux connexions métalliques, le «drain» et la «source», grâce à une grille en polysilicium (à l'origine en métal, d'où son nom).

Si l'on observe la photo d'un MOS prise au microscope électronique (fig. 6.1), on aperçoit un trou rectangulaire entre le drain et la source, que l'on appelle la *diffusion*. A ses deux extrémités, on voit le drain et la source du transistor connectés par des contacts ronds à des fils métalliques. Sur la gauche de la photo apparaît une connexion métallique qui contacte la grille du transistor. Celle-ci descend dans le trou de diffusion pour remonter de l'autre côté afin de séparer complètement le drain et la source. Les transistors de type N-MOS ne conduisent que si la grille est à haute tension. Si elle est à la masse, le transistor est bloqué et aucun courant ne circule. On obtient alors un interrupteur commandé par la grille.

Pour réaliser des fonctions utiles dans un circuit électronique, il faut plusieurs transistors interconnectés. Ces fonctions sont dites «logiques», car on utilise une numérotation binaire, la haute tension étant symbolisée par «1» et la masse par «0».

Si la source d'un transistor MOS est connectée à la masse («0») et que l'on applique un «1» sur la grille, il est conducteur et son drain prend la valeur «0». Si au contraire la grille est «0», le transistor est bloqué. Son drain n'est pas tiré à «0» et pourrait rester à «1». Il est ainsi possible de réaliser des fonctions logiques ET et OU, à la base de tout circuit électronique (chap. 1, logique de Boole) à partir de ces transistors.

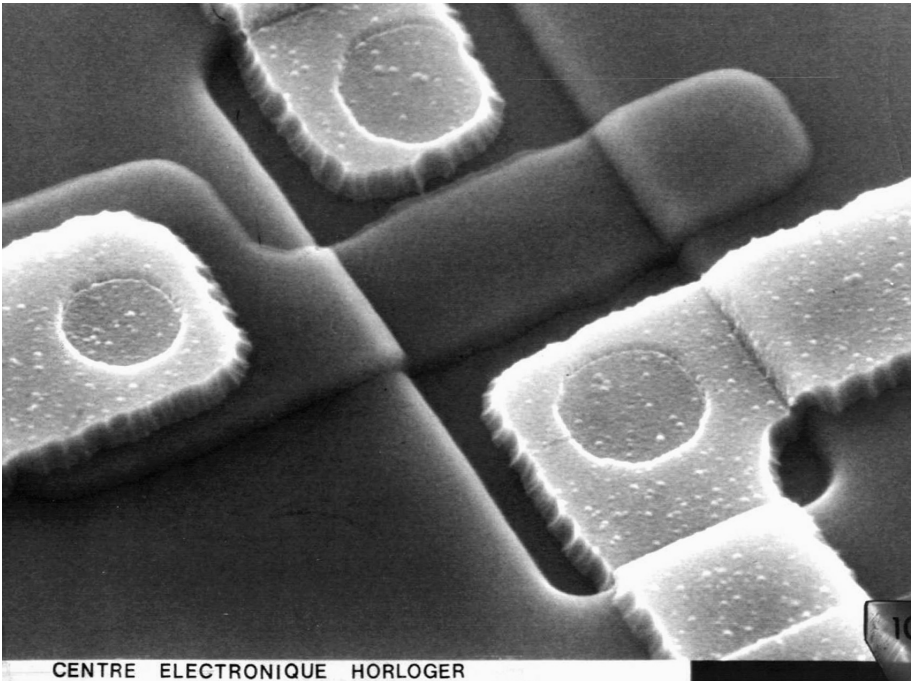


Fig. 6.1 Un transistor MOS photographié au microscope électronique.

L'invention du microprocesseur

Le premier microprocesseur, Intel 4004, est inventé en novembre 1971 par la compagnie américaine Intel.

Ce microprocesseur travaille avec des données de 4 bits, comporte 2300 transistors dans une technologie dite P-MOS, peut adresser une mémoire de programme comportant 4000 mots de 8 bits et additionner deux données de 4 bits en 10 microsecondes avec une horloge de base de 100 kHz [3]. Le 4004 a sensiblement la même puissance de calcul que l'ENIAC en 1946. L'unité de calcul des ordinateurs était jusqu'alors réalisée avec divers composants assemblés sur une carte et le microprocesseur permet de réaliser l'ensemble de cette unité de calcul en un seul circuit intégré, ce qui a pour conséquences directes de diminuer les coûts de façon importante, et surtout d'augmenter extraordinairement les performances en vitesse.

Intel, aujourd'hui célèbre pour ses microprocesseurs Pentium et Itanium, est fondée en septembre 1968 dans la «Silicon Valley» en Californie.

En avril 1969, Intel accepte de développer un circuit pour le compte de la compagnie japonaise Busicom, désireuse de fabriquer des calculatrices de table. Les spécifications de ce circuit sont discutées, et Ted Hoff de Intel propose la réalisation d'une unité de calcul à la fois simple et universelle. D'abord réticente, Busicom accepte finalement la suggestion de Hoff, et charge son ingénieur Masatoshi Shima de rédiger les spécifications du 4004. En avril 1970, Frederico Faggin rejoint Intel et participe à la conception du circuit. Il réalise le dessin des transistors, et constitue à ce titre le personnage clé de la réussite de l'entreprise. Un layout consiste à dessiner les transistors à partir de règles très précises concernant l'espacement et la largeur des conducteurs en métal ou en poly (silicium polycristallin) utilisés. Le layout reproduit à la figure 6.2 ne représente qu'un seul de ces transistors; le 4004 en comportera 2300, que Faggin a au préalable dessinés et interconnectés manuellement. La figure 6.3 offre une vue au microscope électronique de quelques transistors MOS interconnectés.

Début 1971, le prototype de ce microprocesseur 4-bit 4004 est fonctionnel, et, après de longs débats, Intel décide d'en faire un produit standard en novembre de cette

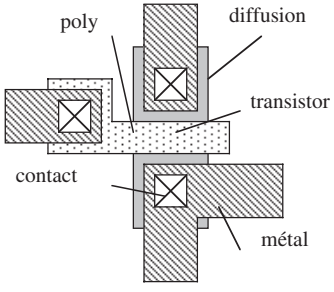


Fig. 6.2 Dessin de layout du transistor MOS vu sous microscope électronique à la figure 6.1.

même année. Mais à cette date, le concept est déjà vendu à Busicom par les dirigeants d'Intel, alors loin d'imaginer l'énorme potentiel dont le microprocesseur est porteur. Heureusement pour eux, Busicom veut négocier le prix des puces à la baisse, offrant du même coup une occasion inespérée de rediscuter l'accord. Intel propose la commercialisation sous son nom du 4004 (exception faite des calculatrices) en échange d'une diminution de prix, offre que Busicom accepte. Quelques années plus tard, le Japon se lancera dans une course effrénée aux microprocesseurs.

Intel développe presque simultanément le premier microprocesseur 8 bits.

En décembre 1969, la compagnie américaine CTC (qui deviendra par la suite Datapoint) charge Intel de développer un circuit pour terminal d'ordinateur. Ses spécifications sont mises au point par un ingénieur de CTC, Vic Poor. Elles comportent qua-

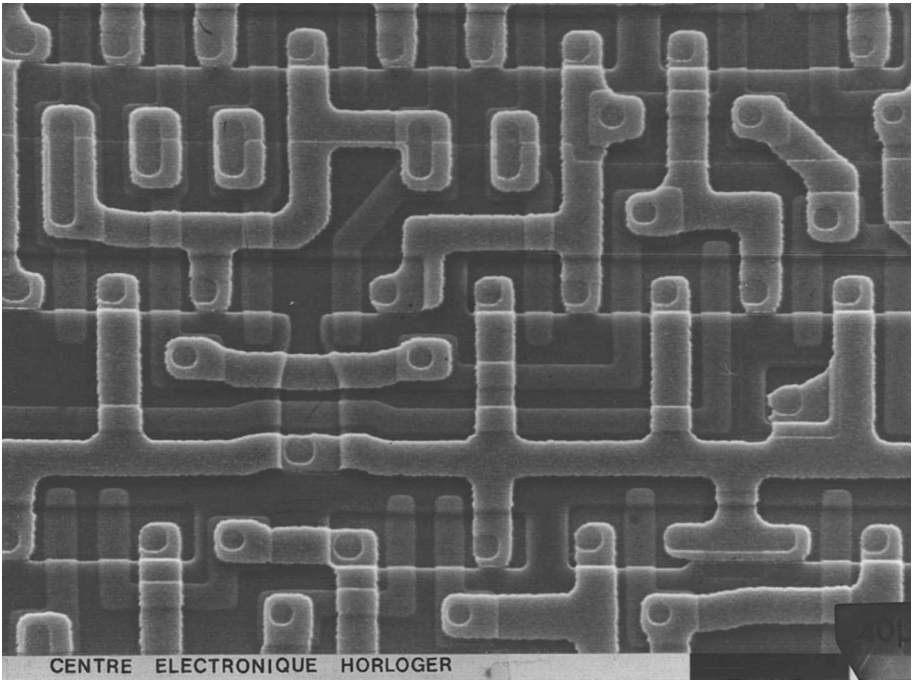


Fig. 6.3 Ensemble de transistors interconnectés (vue en microscopie électronique).

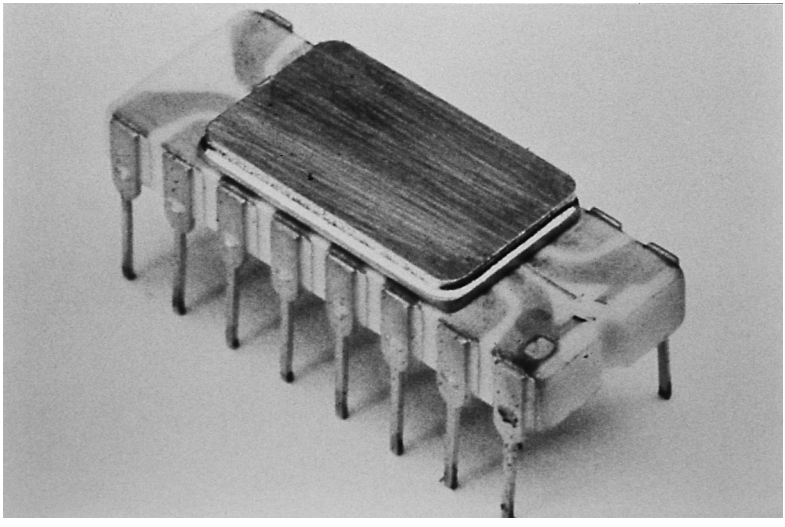


Fig. 6.4 Microprocesseur 4004 encapsulé dans son boîtier (Intel Corp.).

siment le répertoire d'instructions et l'architecture du microprocesseur 8-bit 8008, que Intel annonce en avril 1972, comportant 3500 transistors N-MOS et cadencé à 200 kHz.

CTC demande aussi à Texas Instruments de réaliser ce même circuit, mais ni l'un ni l'autre ne parviendront à satisfaire aux spécifications demandées. La société devra finalement réaliser ce produit à partir de plusieurs circuits assemblés sur une même carte.

En 1970, alors que les deux microprocesseurs 4004 (fig. 3.10 et fig. 6.4) et 8008 (fig. 6.5) sont en phase de conception, on considère que 1000 transistors constituent une limite au nombre total de transistors sur un circuit intégré en technologie de 10 micromètres. Le 4004 et le 8008 en contiennent davantage, démontrant ainsi qu'ils représentent l'un et l'autre les limites de la technologie de l'époque.

Le service juridique d'Intel estime que placer une unité de calcul sur un seul circuit intégré constitue quelque chose de si évident qu'un brevet sur le microprocesseur n'aurait aucune valeur. Intel publie donc les spécificités du 4004 en mars 1970 sans brevetage préalable. Les premières demandes de brevets déposées par d'autres sociétés ne tarderont pas.

Gilbert Hyatt, fondateur de la petite compagnie Microcomputer Inc (financée par Noyce et Moore, les fondateurs de Intel, et disparue en 1970) dépose une demande de brevet en décembre 1970. En 1996, un brevet est accordé à Gary Boone, de Texas Instruments, pour le microprocesseur TMS100.

Qui, de Ted Hoff et Frederico Faggin d'Intel, Masatoshi Shima de Busicom, Vic Poor de CTC, Gilbert Hyatt, et Gary Boone, est le véritable inventeur du microprocesseur? Vingt-cinq ans après l'apparition du 4004 [4], la question demeure en suspens.

Plus récemment, la compagnie Four-Phase Systems de Lee Boysel a revendiqué la paternité du microprocesseur, arguant du fait qu'elle avait réalisé une unité de calcul 8-bit sur un seul circuit intégré appelé «AL1», inspiré de l'architecture de l'IBM 360, et ceci près de 2 ans avant que Intel ne produise son premier microprocesseur.

Le AL1 est surprenant à plus d'un titre, et très différent des 4004 ou 8008 de Intel; il comporte notamment des instructions longues de 32-bit et il est 20 fois plus rapide que le 8008. S'il avait été aussi célèbre que les 4004 et 8008, l'évolution des microprocesseurs aurait probablement été bien différente, notamment en adoptant immédiatement les instructions de 32 bits.

Les premiers microprocesseurs

Lorsque le microprocesseur apparaît, seuls ses concepteurs croient en son potentiel. Ted Hoff et Frederico Faggin, associés à Regis McKenna, un gourou du marketing, vont créer le marché et déclarer «qu'une nouvelle ère électronique est née» [2].

Toute une série de microprocesseurs voient en effet le jour dans les années suivantes [5, 6]:

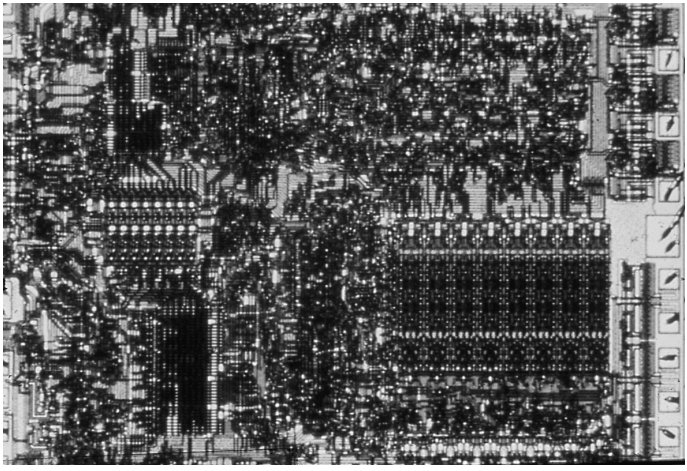


Fig. 6.5 Circuit du microprocesseur 8008 (Intel Corp.).

- en 1971, le TI 1795 (Texas Instruments) jamais commercialisé;
- en 1972, le TI TMS 100;
- en 1973, le Rockwell PPS-4;
- en 1973, le Motorola 6800, dont le démarrage fut difficile (avec puce trop grosse et mauvais rendements);
- en 1974, le TI TMS 1000, premier micro-ordinateur sur puce, doté de mémoires de programme et de données;
- en 1974, le RCA 1802, premier microprocesseur en technologie CMOS, aujourd'hui mondialement répandue;
- en 1974, l'Intel 8080, premier grand succès de Intel, réalisé par F. Faggin et Masatoshi Shima;
- en 1976, le Zilog Z80 de la compagnie Zilog fondée en 1974 par F. Faggin; c'est le microprocesseur 8-bit le plus vendu au monde;
- en 1976, le TMS 9900;
- en 1976, l'Intel MCS-48, premier micro-ordinateur de Intel; il présente des mémoires de programme et de données sur la puce;

Bien d'autres apparaissent ensuite, issus des laboratoires des compagnies AMD, Signetics, National, NEC, Gould, Fairchild et Mostek [5].

Les principales caractéristiques de ces premiers microprocesseurs sont les suivantes: quelques milliers de transistors dans une technologie de 6-10 microns de finesse de trait, une horloge d'environ 1 à 2 MHz, de 50 à 100 instructions et une consommation proche de 1 Watt.

Le concept d'architecture de ces premiers microprocesseurs est le recours à un petit nombre de transistors (quelques milliers) et à une mémoire extérieure contenant à la fois instructions et données (type Von Neumann, chap. 4, fig. 4.19). Les données étant généralement de 8-bit, les instructions logées dans cette même mémoire doivent avoir ce même format. Mais toutes les instructions ne peuvent être codées sur 8-bit, certaines étant nécessairement plus longues. La question sera résolue par l'emploi des instructions de plusieurs octets, lus successivement dans la mémoire externe. La figure 6.6 illustre un exemple de trois instructions composées de un, deux et trois octets.

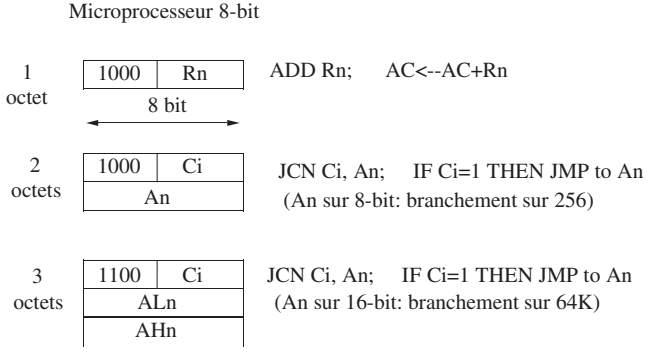


Fig. 6.6 Trois instructions dites «multi-octets» ou «multi-bytes».

L'usage d'instructions composées de plusieurs octets a diverses conséquences sur l'architecture du microprocesseur.

Celui-ci doit tout d'abord savoir reconnaître la longueur de chacune de ces instructions de longueur variable comportant un, deux, trois ou davantage d'octets, afin notamment de pouvoir identifier le début de l'instruction suivante. Autre conséquence, la lecture et l'exécution de l'instruction complète demandent un nombre important d'étapes, afin de pouvoir lire successivement les différents octets qui composent cette instruction.

☞ La figure 6.7 représente l'architecture d'un microprocesseur 8-bit avec une mémoire externe comportant instructions et données. Pour exécuter une instruction, on charge tout d'abord le compteur de programme, qui pointe ensuite l'instruction dans le registre d'adresse de la mémoire. L'instruction est lue, puis chargée dans le registre d'instructions et décodée. Ce décodage indique alors s'il faut aller chercher un ou deux octets supplémentaires en incrémentant le compteur de programme et en lisant le second et le troisième octet. Pour l'exécution proprement dite, il faut donner l'adresse de l'opérande en mémoire, le lire, puis le charger dans le banc de registres ou dans l'unité arithmétique et logique (ALU) pour effectuer un calcul. Le résultat se trouve alors dans l'accumulateur (Accu) et doit être sauvé dans un registre ou dans la mémoire à une nouvelle adresse. Tous ces transferts utilisent le même

bus, et exigent par conséquent de nombreuses étapes. L'unité de séquencement de l'instruction, très complexe, occupe pour cette raison les trois-quarts de la surface de la puce. Ce format d'instructions est donc très différent de celui des premiers ordinateurs comportant des instructions longues sur un seul mot et décrites au chapitre 4.

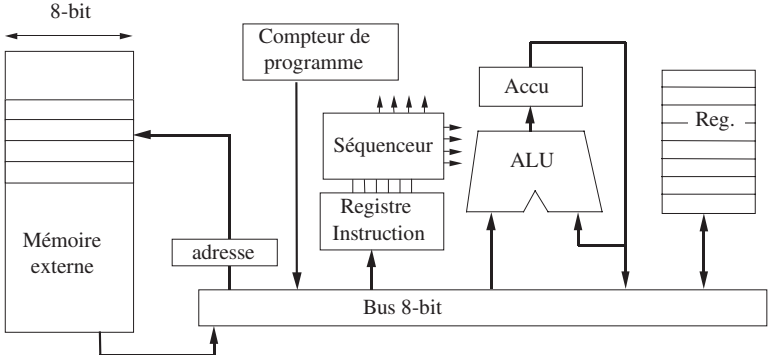


Fig. 6.7 Architecture d'un microprocesseur 8-bit.

Les microprocesseurs microprogrammés

L'année 1978 marque le début de l'ère des microprocesseurs 16-bit. Les répertoires d'instructions deviennent plus complexes, tout en conservant leurs formats de plusieurs octets afin de demeurer compatibles avec les logiciels existants.

L'Intel 8086 apparaît tout d'abord, suivi une année après du Zilog Z8000 et du Motorola 68'000, et de nombreux autres [7]. Composés de quelques dizaines de milliers de transistors de 3 à 4 microns de finesse de trait, ces microprocesseurs fonctionnent à 10 MHz, et comportent de 150 à 200 instructions complexes.

En 1979, la lutte devient très compétitive entre Intel, Motorola et Zilog. A la fin de l'année, Intel se retrouve largement distancé par Motorola et Zilog [1]. Plutôt que de se lancer dans une course désespérée en tentant de réaliser en quelques mois un nouveau microprocesseur, Intel mise alors sur l'aspect système, avec notamment les coprocesseurs, les logiciels et les outils de développement. Cette réorientation, appelée CRUSH, soutenue par une vaste campagne de publicité, rencontre un large succès, et permet bientôt à Intel de reprendre 85% des parts de marché des 16-bit. Le 8086 demeure à la base de tous les futurs microprocesseurs de Intel, 186, 286, 386, 486, Pentium et Pentium Pro, III et IV, qui exécutent encore aujourd'hui du code x86. Ce succès est en partie dû à IBM, qui choisit le 8088 (un 8086 avec un bus externe de 8-bit au lieu de 16-bit) pour équiper ses PC. Le Motorola 68'000 quant à lui est choisi par Apple pour équiper ses Macintosh.

Un tournant s'amorce alors dans la réalisation des microprocesseurs, dont le nombre et la complexité des instructions nécessitent une unité de décodage et de séquencement toujours plus puissante et plus complexe…

Ces unités sont jusque-là réalisées par un assemblage de portes logiques comportant quelques transistors et éléments de mémorisation (logique câblée). Le microprocesseur Z8000 atteint ainsi les limites de complexité de l'unité câblée de décodage et de séquencement des instructions, qui occupe les deux tiers de la puce. Elle est la dernière conçue de cette manière, c'est-à-dire inspirée de machines plus simples comme les INTEL 8085 et Z80. Les autres processeurs 16 bits de complexité comparable au Z8000 sont par la suite élaborés à partir de techniques proches de la microprogrammation [2], décrites par Wilkes (chap. 4, fig. 4.33).

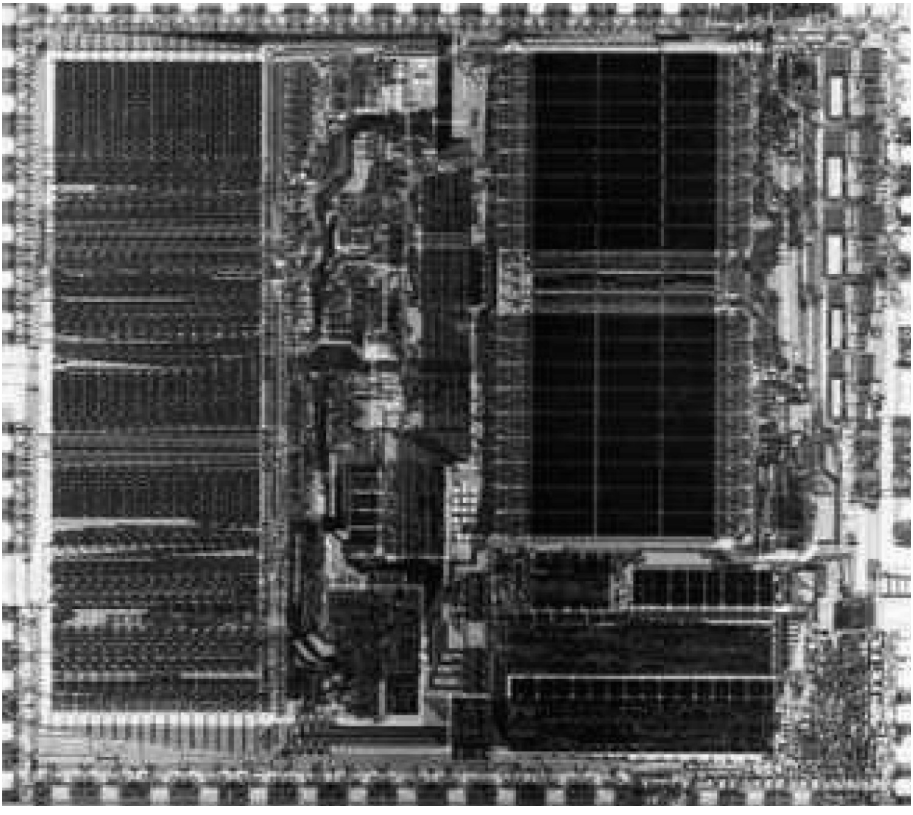


Fig. 6.8 Circuit du microprocesseur 68'000 (Motorola Inc.).

Le Motorola 68'000 est un microprocesseur 16 bits microprogrammé comportant une unité de décodage et de séquencement des instructions constituée par une mémoire

morte (ou ROM pour «*read-only memory*», soit une mémoire pouvant être seulement lue) (fig. 6.8). Chaque instruction est exécutée par un petit programme (ou microprogramme) qui réalise toutes les étapes nécessaires à l'exécution de l'instruction.

☞ Le principe de fonctionnement d'une telle unité de commande est illustré à la figure 6.9. L'instruction est tout d'abord chargée dans le registre d'instructions (RI) après la lecture en mémoire. Un transcodeur identifie l'adresse du début du microprogramme correspondant à cette instruction, avant que les micro-instructions, adressées tour à tour par un micro-compteur de programme (μPC), ne soient exécutées l'une après l'autre pour indiquer les opérations à faire à l'unité de calcul.
Cette manière de procéder rend les modifications du microprogramme plus aisées par rapport à une unité câblée. L'unité de commande du 68'000 occupe toutefois près de 70% de la surface de la puce, et le nombre d'étapes nécessaires à l'exécution d'une seule instruction, soit tout un microprogramme, demeure élevé et ne permet pas l'obtention de grandes performances.

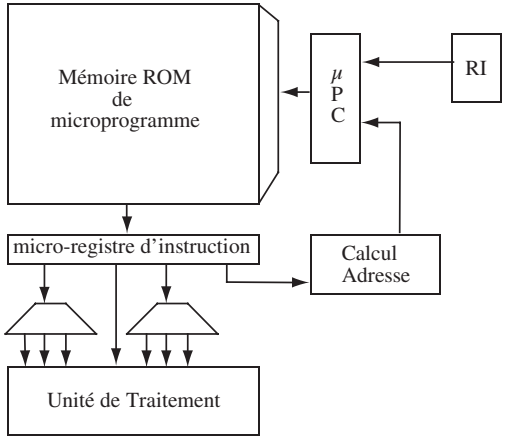


Fig. 6.9 Principe d'une unité de commande microprogrammée.

Les microprocesseurs RISC

Les constructeurs de microprocesseurs vont réaliser des machines de plus en plus complexes, présentant des répertoires d'instructions pouvant en compter entre 300 et 500: les CISC (Complex Instruction Set Computer).

A la fin des années 1970, on tente de réaliser en matériel ce qui existe en logiciel, notamment des procédures de systèmes d'exploitation ou de langages de haut niveau, ou d'exécution directe de ces langages sans passer par un compilateur. Mais les microprocesseurs nécessaires à une telle entreprise sont si complexes que ce projet est abandonné.

Après analyse de programmes produits par un compilateur, force est de constater que près de la moitié des instructions du répertoire d'une machine CISC ne sont

jamais utilisées; certaines instructions complexes peuvent d'ailleurs être remplacées par un ensemble de quelques instructions simples sans affecter la vitesse, des instructions simples étant plus rapides à exécuter.

C'est à partir de ces constatations que naît le concept des machines RISC (Reduced Instruction Set Computer) [8].

La première de ce type à voir le jour est l'IBM 801 (1975), avant même les célèbres RISC I et II de Berkeley (1980) qui feront connaître le concept [10]. IBM avait par ailleurs été la première compagnie à proposer une machine CISC, l'IBM 360 (1964) [9].

Au début des années 1980, le débat RISC-CISC fait rage. Les adversaires des RISC avancent qu'elles ne peuvent pas effectuer des opérations en virgule flottante ni comporter de mémoire virtuelle, et soutiennent que c'est la centaine de registres équipant la machine qui explique l'augmentation de ses performances, et non pas son répertoire d'instructions limité. D'après eux, augmenter le nombre de registres d'une machine CISC est le meilleur moyen d'obtenir le microprocesseur le plus puissant. Argument réfuté par leurs concurrents, pour qui la possibilité de loger un grand nombre de registres dans un microprocesseur est essentiellement liée à la faible taille de l'unité de séquencement et de décodage des instructions (elle n'occupe pas les 3/4 de la puce comme chez les CISC, ce qui laisse de la place pour les nombreux registres). Autre critique, la longueur des programmes RISC: il faut en effet 2 à 3 instructions RISC pour obtenir l'équivalent d'une seule instruction CISC. Il est toutefois plus difficile d'écrire un bon compilateur pour un CISC de 300 instructions, que pour un RISC ne comportant que 50 à 100 instructions simples. Les machines pionnières RISC I et RISC II présentent un format d'instructions bien différent de celui des CISC (fig. 6.10), à savoir des instructions longues et sur un seul mot. Ce format simplifie sensiblement la conception des unités de commande et explique leur faible taille.

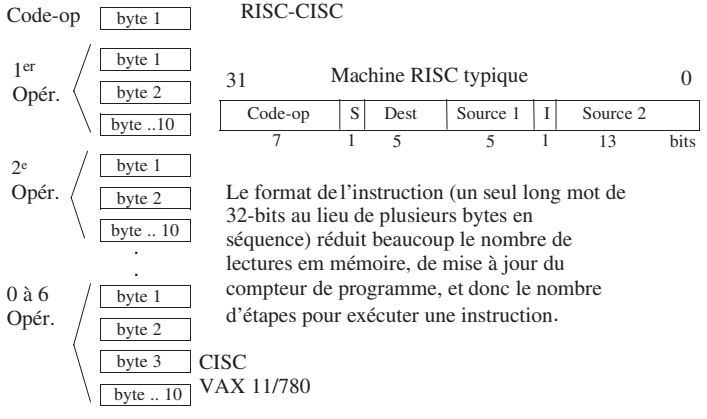


Fig. 6.10 Formats des instructions CISC et RISC.

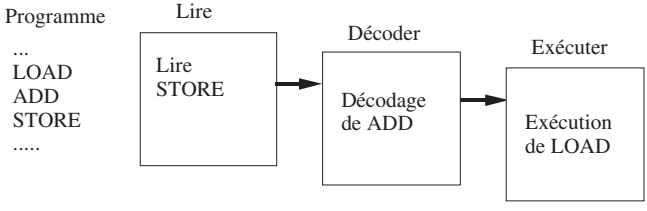
La taille ne représente plus que 6% de la surface de la puce pour le RISC I, 10% pour le RISC II, alors que les unités de commande des machines CISC occupaient 75% de la surface de la puce.

La machine RISC I comporte 138 registres de 32 bits et, comme le RISC II, est aussi performante que les machines CISC. La preuve est ainsi faite qu'une machine simple, avec peu de transistors et de connexions, est plus rapide qu'une machine complexe.

Le fonctionnement en pipeline est une autre caractéristique importante des machines RISC. Cette technique, déjà été utilisée en 1964 par IBM pour le IBM 360/91, ressemble à une chaîne de montage dans laquelle les instructions sont à divers stades de leur exécution.

☞ La figure 6.11 donne une illustration d'un tel fonctionnement à trois étages: le premier lit les instructions en mémoire, le second les décode tandis que le troisième les exécute. Ainsi, avec un programme comportant par exemple LOAD, suivi de ADD et de STORE, lorsque LOAD est exécuté dans le 3e étage, le 2e décode ADD pendant que le premier lit la 3e instruction STORE.

Les performances offertes par un tel fonctionnement sont bien supérieures à celles d'une machine devant attendre qu'une instruction soit terminée pour commencer à lire la suivante.



Le microprocesseur est décomposé en trois étages de pipeline (comme une chaîne de montage); en même temps, on exécute le LOAD, on décode le ADD et on lit en mémoire le STORE.

Fig. 6.11 Pipeline.

☞ La figure 6.12 illustre la chronologie du fonctionnement d'une machine pipeline. L'instruction LOAD est lue dans un premier cycle d'horloge et décodée pendant le suivant, pendant que la 2e instruction ADD est lue. Lors d'un 3e cycle d'horloge, l'exécution de LOAD se termine, ADD est décodé et la lecture de la 3e instruction STORE commence.

Les performances de la machine augmentent ainsi proportionnellement au nombre d'étages pouvant exécuter simultanément des instructions.

La technique pipeline présente toutefois un problème de branchement (fig. 6.13). Lorsqu'une instruction de branchement est par exemple décodée dans le 2e étage, l'instruction suivante est quant à elle déjà dans le pipeline, alors que si le branchement est pris, elle ne devrait pas être exécutée. Il faut alors la supprimer, ou utiliser d'autres techniques plus sophistiquées, qui augmentent la complexité du microprocesseur et qui diminuent d'autant le gain de performances offert par le pipeline.

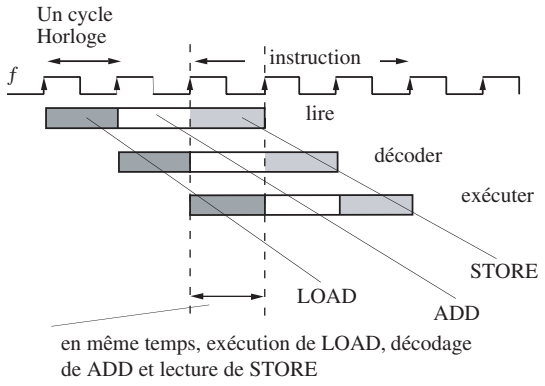


Fig. 6.12 Déroulement dans le temps des opérations d'une machine pipeline.

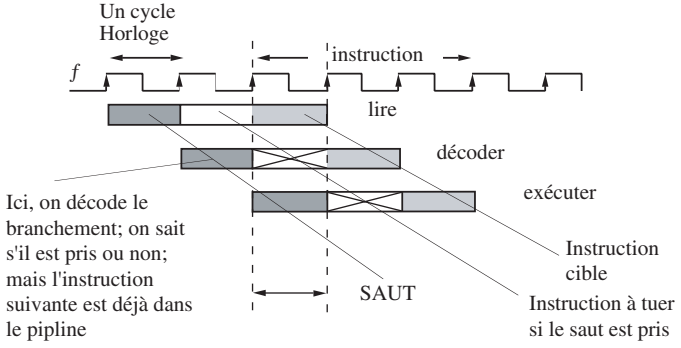


Fig. 6.13 Problème du branchement dans les pipelines.

Les microprocesseurs modernes

Les microprocesseurs modernes, tels que Pentium, Itanium, Alpha, PowerPC ou MIPS, sont différents des machines RISC des années 1985. Leur fonctionnement repose sur un pipeline, mais ils contiennent plusieurs unités d'exécution afin de traiter plusieurs instructions en parallèle.

La figure 6.14 illustre un tel fonctionnement où deux instructions sont exécutées simultanément. Le microprocesseur comporte dans cet exemple trois unités d'exécution en parallèle, deux unités de calcul et une unité de branchement. Ce type de microprocesseur est dit «superscalaire».

Cette parallélisation présente toutefois des limites, notamment lorsque deux instructions dépendent l'une de l'autre, ce qui se produit:

- lors d'une addition, suivie d'une autre dont le premier opérande est le résultat intermédiaire de la première;
- lors d'une comparaison, suivie d'un branchement dépendant du résultat de celle-ci.

On ne peut exécuter ces instructions en parallèle, mais en série seulement, et il faut accepter la perte d'efficacité du microprocesseur qui en résulte. Un programme comportant beaucoup de ces dépendances entre instructions limitera de façon notable les performances d'une machine superscalaire.

Le microprocesseur est capable de décider seul s'il peut exécuter ou non deux ou trois instructions en parallèle, si elles sont dépendantes ou non, et à quelles unités d'exécution il doit les envoyer. Ce potentiel n'existe bien entendu qu'au prix d'une unité de décodage très complexe.

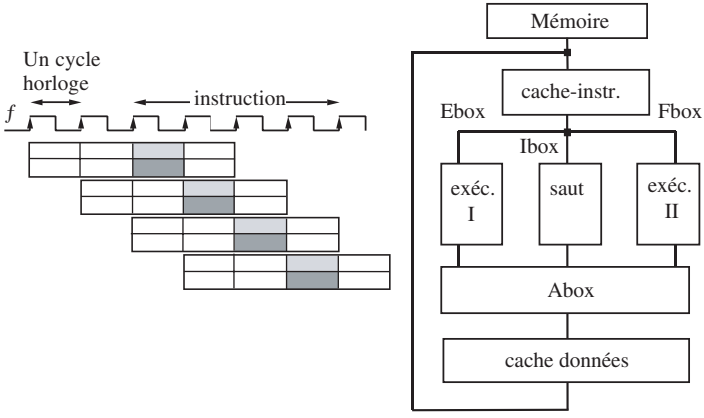


Fig. 6.14 Microprocesseur superscalaire.

Les unités de commande des machines superscalaires étant devenues à nouveau très complexes, une nouvelle simplification s'impose.

Ainsi apparaît une nouvelle architecture de microprocesseurs, appelée VLIW (*Very Large Instruction Word*) ou EPIC (*Explicitely Parallel Instruction Computing*). Le compilateur est ici chargé d'effectuer tout le travail complexe, c'est-à-dire de détecter les dépendances entre instructions et les unités d'exécution auxquelles elles doivent être adressées. L'unité de commande peut ainsi se reposer sur le compilateur, et demeurer considérablement plus simple que dans les machines superscalaires.

L'architecture VLIW, illustrée à la figure 6.15, est très similaire à celle d'une machine en pipeline. La différence réside dans l'exécution non pas d'une seule opération, mais de plusieurs, spécifiées dans une très large instruction.

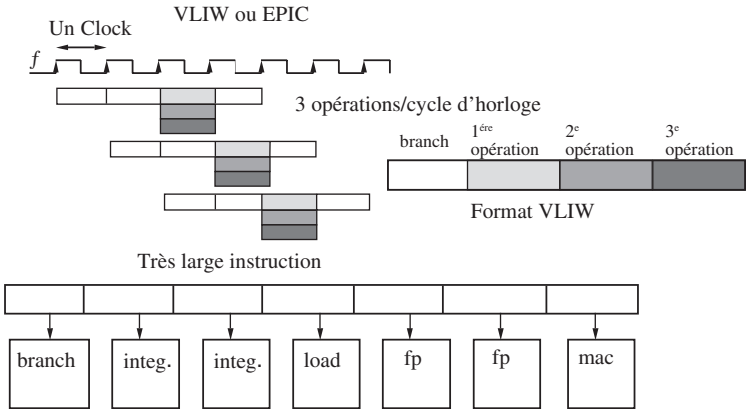


Fig. 6.15 Architecture VLIW ou EPIC.

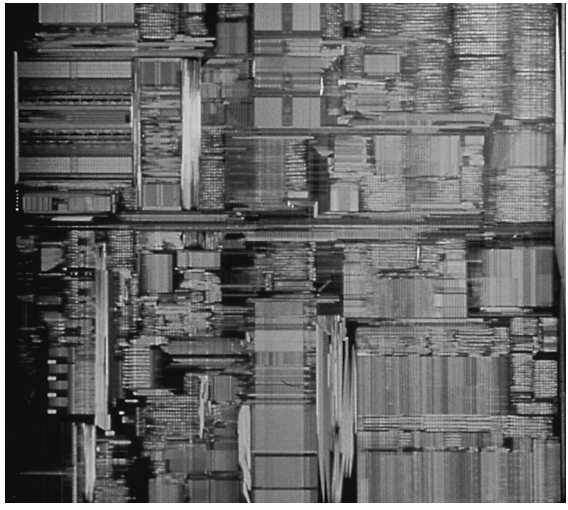


Fig. 6.16 Circuit Intel Pentium (1992) (Intel Corp.).

Les nouveaux microprocesseurs de Intel, l'Itanium et le McKinley, ont adopté ce type d'architecture, développée en collaboration avec Hewlett-Packard. Les performances ne sont toutefois pas aussi excellentes qu'on l'imaginait il y a quelques années, ce qui conduit notamment Intel à pousuivre sa production de Pentium superscalaire (fig. 6.16).

Les mémoires

Si les microprocesseurs atteignent aujourd'hui des fréquences de plusieurs GigaHertz, les mémoires ont quant à elles de plus en plus de difficulté à atteindre de telles fréquences.

Les tailles toujours plus grandes des mémoires les rendent toujours plus lentes. Si un processeur va par exemple 10 fois plus vite que la mémoire principale, et s'il y est directement branché, il ne pourra lire ses instructions qu'à la vitesse offerte par la mémoire. Pour remédier à cela, on emploie des mémoires cache (fig. 6.17), que l'on insère entre les deux éléments, et dans lesquelles on charge une petite partie du programme. Une mémoire cache étant de petite taille, sa vitesse est aussi importante que celle du microprocesseur; il peut toutefois arriver que l'instruction désirée ne soit pas dans la cache (dans 1 à 5% des cas, selon les caches et leurs tailles). Il faut alors arrêter le microprocesseur, aller dans la mémoire principale afin de charger la cache avec les instructions manquantes, et recommencer. Le concept de cache est appliqué par IBM dès le milieu des années 1970.

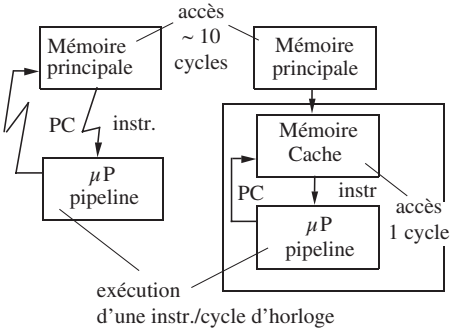


Fig. 6.17 Mémoire cache.

La commercialisation des microprocesseurs

Depuis 1971, les microprocesseurs ont envahi le monde. D'abord sous forme de 8 bits, puis de 16 bits, avant que n'apparaissent les machines RISC 32 bits et les microprocesseurs superscalaires et EPIC de 32 ou 64 bits. Malgré cette évolution vertigineuse, les 8 bits demeurent très employés aujourd'hui encore.

Certains microprocesseurs ont atteint une puissance importante, en utilisant des mots de 64 bits et des dizaines de millions de transistors en technologie 0,13 micron et cadencés à 2 GHz (soit 20 000 fois plus rapide que le 4004). Ces microprocesseurs «haut de gamme» constituent aujourd'hui le cœur des ordinateurs, des plus petits PC aux plus gros ordinateurs parallèles. Le marché des microprocesseurs de moindre puissance comme les 8 bits reste plus important encore [11], ceux-ci rentrant dans la

conception des voitures, des appareils électroménagers, et de tout appareil portable. Les principales compagnies de microélectronique ont conçu leurs propres microprocesseurs, comme Intel, Motorola, Texas Instruments, IBM, Digital, Zilog, Sun, Hitachi ou ARM.

En Suisse, le CSEM et XEMICS réalisent des microprocesseurs 8 bits à très basse consommation, utilisés notamment pour les montres électroniques [14]. L'un d'eux, le CoolRISC, est conçu sur le principe du pipeline, et peut ainsi exécuter une instruction par cycle d'horloge comme un microprocesseur RISC. Grâce aux améliorations apportées à l'architecture des microprocesseurs et aux possibilités offertes par les dernières technologies des circuits intégrés (0,25 µm de finesse de trait), la consommation d'énergie par instruction d'un CoolRISC 8 bits actuel est d'environ 400 000 fois inférieure à celle d'un 4004 en 1971. La figure 6.18 offre une vue du CoolRISC 8 bits en technologie 1 mm de finesse de trait. Les motifs réguliers aux allures de bandes horizontales sont des cellules accolées les unes aux autres et interconnectées automatiquement.

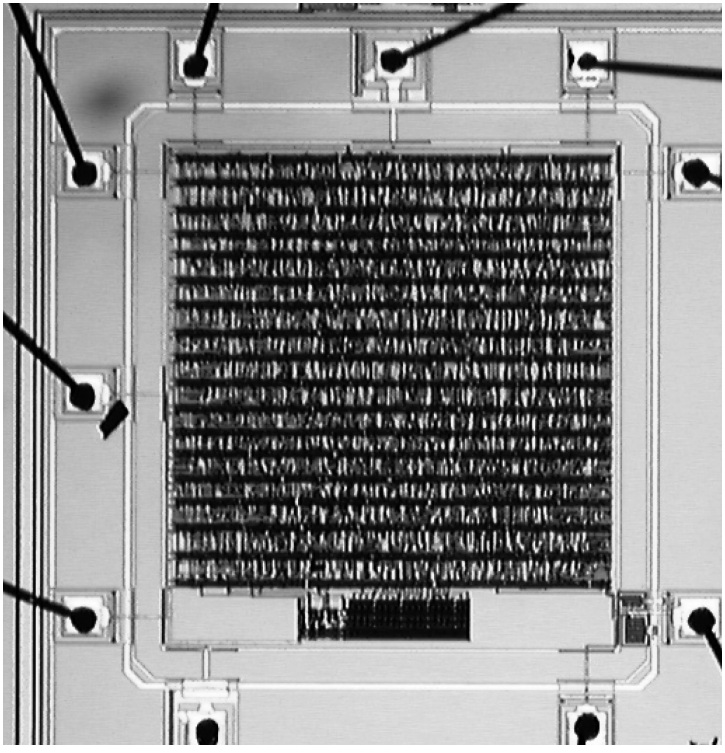


Fig. 6.18 Microphotographie du CoolRISC 8 bits en 1 mm, 1995.

L'impact des ordinateurs et des microprocesseurs dans le monde est croissant. Ils sont aujourd'hui omniprésents, dans notre vie professionnelle ou dans nos loisirs, notamment avec les jeux sur ordinateurs ou, plus récemment, avec le développement de l'Internet. Il s'est vendu en Suisse en 1998 près de 1 million d'ordinateurs personnels.

Andrew Grove, né en 1936, PDG et fondateur de Intel, a été élu en 1997 «man of the year» par le *Time Magazine* [12]. En effet, 90% des ordinateurs vendus dans le monde cette année-là étaient équipés d'un microprocesseur Intel. La firme Intel a réalisé en 1996 un chiffre d'affaires de 30 milliards de francs suisses!

Les performances des microprocesseurs

Les microprocesseurs ont vu leurs performances s'accroître de manière stupéfiante ces dernières années, comme l'illustre la table ci-dessous:

1971	4004	0,0015 MIPS
1972	8008	0,003 MIPS
1975	8080	0,028 MIPS
1978	8086	0,570 MIPS
1982	80286	1,3 MIPS
1985	80386	2,2 MIPS
1989	80486	8,7 MIPS
1993	Pentium	64 MIPS
1997	Pentium II	320 MIPS
1999	Pentium III	710 MIPS
2000	Pentium IV	2500 MIPS
2001	Itanium	2800 MIPS

(MIPS: Million d'instructions exécutées par seconde)

Le taux de croissance des performances des microprocesseurs est donc de près de 60% par an. Les tailles des mémoires de données (DRAM), utilisées dans les ordinateurs personnels, ont quant à elles été multipliées d'un facteur 4 tous les 3 ans entre 1977 et 1992, comme l'illustre la table ci-dessous:

1977	16 Kbit
1980	64 Kbit
1983	256 Kbit
1987	1 Mbit
1989	4 Mbit
1992	16 Mbit
1995	64 Mbit
1999	256 Mbit
2002	1 Gbit
2005	4Gbit

Quelques perspectives

La prévision SIA 2001 donne l'évolution probable des technologies utilisées pour la fabrication des circuits intégrés et des microprocesseurs, de 2001 à 2016.

Le nombre de transistors total pourrait être de 9 milliards en 2016, soit l'équivalent de 90 Pentium sur une même puce. L'accroissement le plus important serait celui des délais de connection et de la longueur de fil sur chacune des puces (près de 24 km en 2016!) (fig. 6.19).

En 2016, la consommation des puces microprocesseurs est prédite comme extrêmement importante, soit 288 watts ou l'équivalent de 5 ampoules de 60 watts sur 3 cm^2. Cette prévision tient en particulier compte d'une tension d'alimentation extrêmement basse de 0,4 volt et du fait que seul un nombre très limité de blocs logiques du circuit travailleront à 28GHz. Si tel n'était pas le cas, la consommation dépasserait le kilowatt, ce qui est manifestement impossible.

Année	2001	2004	2007	2010	2013	2016
technologie	0,15 µm	0,09 µm	0,065 µm	0,045 µm	0,032 µm	0,022 µm
taille puce	3,1 cm^2	3,1 cm^2	3,1 cm^2	3,1 cm^2	3,1 cm^2	3,1 cm^2
transistors	276 M	553 M	1,1 G	2,2 G	4,4 G	8,8 G
fréquence	1,7 GHz	4 GHz	7,5 GHz	11,5 GHz	19,3 GHz	28,5 GHz
tension	1,1 Volt	1,0 Volt	0,7 Volt	0,6 Volt	0,5 Volt	0,4 Volt
consom.	130 watt	160 watt	190 watt	218 watt	251 watt	288 watt
long. métal	0,8 km	1,5 km	3 km	5 km	10 km	28 km

Fig. 6.19 Performances des microprocesseurs de 2001 à 2016.

Références

[1] M. S. Malone, *The Microprocessor. A Biography*, Springer-Verlag, New-York, Inc., published by Telos, Santa-Clara, CA, 1995.

[2] B. L. Peuto, «Mainframe History Provides Lessons», *Microprocessor Report*, March 31, 1997.

[3] F. Faggin, M. E. Hoff, S. Mazor, M. Shima, «The History of the 4004», *IEEE Micro*, December 1996, pp. 10-20.

[4] «Microprocessor Pioneers Reminisce», *Microprocessor Report*, Vol. 5 N° 24, December 26, 1991.

[5] R. G. Daniels, «A Participant's Perspective», *IEEE Micro*, December 1996, pp. 21-31.

[6] D. P. Siewiorek, C. G. Bell, *A. Newell Computer Structures: Principles and Examples*, McGraw-Hill, 1982.

[7] F. Anceau, «The Architecture of Microprocessors», *Micro-electronics Systems Design*, Addison-Wesley, 1986.

[8] D. Tabak, «RISC Architecture», Research Studies Press Ldt, John Wiley & Sons Inc., 1987.

[9] P. Robinson, «How Much of a RISC» BYTE April 1987, pp. 143-150.

[10] D. A. Patterson, C. H. Séquin, «A VLSI RISC» *IEEE Computer*, Vol. 15, N° 9, pp. 8-21, Sept. 1982.

[11] M. Slater, «The Microprocessor Today», *IEEE Micro*, December 1996, pp. 32-44.

[12] TIME Magazine, December 29, 1997 – January 5, 1998, Vol. 150, N° 26, pp. 29-70.

[13] C. PIGUET, «Are Early Computer Architectures a Source of Ideas for Low-Power?», Volta'99, Como, Italy, March 4-5, 1999.

[14] C. PIGUET *et al.*, «Low-Power Design of 8-bit Embedded CoolRisc Microcontroller Cores», *IEEE JSSC*, Vol. 32, N° 7, July 1997, pp. 1067-1078.

[15] Bruno JACOMY, *Une histoire des techniques*, Coll. Points S67, Seuil, 1990.

CONCLUSION

«My interest is in the future, because I am going to spend the rest of my life there.»
Charles F. Kettering, American Inventor

Cette brève histoire, débutée 2 à 3 millénaires avant Jésus-Christ, touche à sa fin. L'évolution des premières machines à calculer vers les ordinateurs actuels semble continue, à peine perturbée par les «sauts» technologiques importants comme l'apparition du transistor en 1947 et celle du circuit intégré en 1958, qui marquent du même coup le début de l'ère électronique.

L'ordinateur électronique est-il la dernière étape de cette évolution? Rien n'est moins sûr. Il y a 20 ou 30 ans, beaucoup d'espoirs avaient été placés dans l'ordinateur optique, utilisant la vitesse de la lumière pour effectuer des opérations logiques. Le projet a depuis été abandonné. Certains évoquent les ordinateurs «biologiques», où l'ADN (acide désoxyribonucléique) serait utilisé comme moyen de stockage de l'information et de calcul.

On parle aujourd'hui beaucoup des nanotechnologies avec différents dispositifs qui pourraient remplacer le transistor et calculer avec des éléments 1000 fois plus petits que les transistors et ne consommant presque pas d'énergie.

Les prévisions faites en 2001 par l'association des fabricants de semiconducteurs comportent pour 2016 quelque 10 à 20 milliards de transistors sur un seul circuit de silicium, avec des finesses de trait de quelque 0,02 micron ou 20 nanomètres, des tensions très basses de l'ordre de 0,4 volt et des consommations atteignant 300 watts pour des microprocesseurs de très hautes performances. Intel, à fin 2001, a même annoncé avoir produit le plus petit transistor jamais construit, de 15 nanomètres (0,015 µm) et capable de commuter à 2000 GHz! Résultat impressionnant de la microélectronique, il y a aujourd'hui davantage de transistors (10^{17}) dans le monde que de fourmis (10^{16})…

Cependant, quel est l'avenir de la microélectronique, des microprocesseurs et des ordinateurs? Sommes-nous proches de la fin de cette histoire étonnante? Est-ce que les nanotechnologies vont remplacer la microélectronique, et proposer des machines à calculer complètement différentes de nos microprocesseurs d'aujourd'hui?

Des nanodispositifs ont déjà été construits et expérimentés, comme le transistor à un seul électron, les nanotubes de carbone, les interrupteurs moléculaires, qui sont tous de dimensions voisines du nanomètre et peuvent plus ou moins être utilisés comme interrupteurs. Quelques fonctions logiques réalisées avec ces éléments ont démontré leur fonctionnement.

Néanmoins, nous ne sommes qu'au stade de l'élément, comme nous l'étions pour le transistor avant l'invention du circuit intégré. La fabrication de certains de ces nanodispositifs est encore au stade d'une construction atome par atome à l'aide d'un microscope à force atomique. D'autres, comme les nanotubes de carbone, sont fabriqués tous ensemble, mais seulement une petite partie sont des dispositifs interrupteurs, les autres étant de simples conducteurs, sans que l'on puisse définir à l'avance ceux qui seront utilisables et ceux qui ne le seront pas. On ne voit pas très bien comment on pourrait alors dessiner à l'avance un plan de construction de ces nano-éléments, comme on le fait aujourd'hui pour les transistors qui sont placés et surtout interconnectés selon un plan extrêmement précis et complexe pour... simplement donner un microprocesseur.

Ainsi, les méthodes de conception utilisant des nanodispositifs pourraient être complètement différentes de ce que nous réalisons aujourd'hui en microélectronique. On entrevoit plutôt des méthodes où l'on construit un nanocircuit pour ensuite chercher s'il réalise une fonction utile ou intéressante. Des principes d'auto-organisation sont aussi possibles, laissant le nanocircuit trouver lui-même la fonction utile ou intéressante. C'est un peu similaire à la sélection naturelle, mais extrêmement loin des méthodes de conception de microprocesseurs actuels.

En conclusion, on peut affirmer que si les méthodes de conception sont à ce point différentes, elles ne pourront pas produire les mêmes résultats. Si l'on arrive à construire avec des nanodispositifs une machine permettant de calculer, celle-ci sera nécessairement extrêmement différente des microprocesseurs actuels. Mais nous ne connaissons pas aujourd'hui la structure d'une telle machine, qui serait alors le maillon suivant de l'histoire du calcul.

ANNEXES

Le programme pgd de Kilburn

Pour comprendre en détail le programme de Kilburn, il faut se rappeler les ressources disponibles à l'époque. Le programmeur a accès à la mémoire ainsi qu'aux deux registres de l'ordinateur que sont le compteur d'instruction CI, chargé de mémoriser l'adresse de l'instruction en cours, et l'accumulateur C, chargé de mémoriser un opérant. Dans ce qui suit on utilise les notations:

CI:	compteur d'instruction
C:	accumulateur
@n:	mot en mémoire à l'adresse n

pour désigner ces deux registres respectivement pour désigner un mot de 32 bits stocké en mémoire à l'adresse n.

Le programmeur peut en outre compter sur un jeu d'instructions fait de 7 instructions distinctes. Quatre instructions sont destinées au contrôle du séquencement des instructions. Ce groupe comprend des sauts inconditionnels (000,100) et conditionnel (011), ainsi qu'une instruction stop (111) pour arrêter le programme. Les 3 instructions restantes commandent la soustraction (x01) ainsi que les transferts entre mémoire et accumulateur (010,110). La liste complète des 7 instructions est reportée dans la table.

Les tableaux de la page suivante sont une transcription du programme de Kilburn au moyen des notations définies plus haut. On y trouve le programme qui occupe la mémoire entre les adresses 1 à 19 et une deuxième partie occupant les adresses 20 à 27 destinées aux constantes et variables. Kilburn a porté les instructions dans la colonne de gauche sous une forme symbolique, alors que les colonnes de droite contiennent le code binaire correspondant aux champs code opératoire et opérant de l'instruction.

opération	opcode	commentaire
@n->CI	000	saut en @n
CI+@n->CI	100	saut relatif de @n
-@n->C	010	C:=-@n
C->@n	110	@n:=C
C-@n->C	x01	C:=C-@n
if C<0 CI+1->CI	011	saut cond. de la prochaine instr.
stop	111	stop

instruction	C	25	26	27	ligne	code 01234	111 345	remarque
-@24->C	$-b_1$	-	-	-	1	00011	010	
C->@24			$-b_1$		2	01011	110	
-@26->C					3	01011	010	
C->@27				b_1	4	11011	110	
-@23->C	a				5	11101	010	C:=a
C-@27->C	$a-b_n$		$-b_n$	b_n	6	11011	001	C:=C-b
test					7	-	011	skip if neg
CI+@20->CI					8	00101	100	jmp 6
C-@26->C	t_n				9	01011	001	C:=C+b
C->@25		t_n			10	10011	110	temp:=C
-@25->C	$-t_n$				11	10011	010	C:=-temp
test					12	-	011	skip if neg
stop					13	-	111	stop
-@26->C	b_n				14	01011	010	C:=b
C-@21->C	b_{n+1}				15	10101	001	C:=C-1
C->@27	b_{n+1}			b_{n+1}	16	11011	110	b:=C
-@27->C	$-b_{n+1}$				17	11011	010	
C->@26			$-b_{n+1}$		18	01011	110	
@22->CI					19	01101	000	jmp 4

adr	val
20	-3
21	1
22	4

adr	val
23	-a
24	b

adr	val
25	temp
26	$-b_n$
27	b_n

Lectures suggérées

En français:

Georges IFRAH, *Histoire universelle des chiffres*, Bouquins, Robert Laffont, 1994.

Georges JEAN, *L'écriture, mémoires des hommes*, 24/Découvertes Gallimard/archéologie, 1987.

Samuel Noah KRAMER, *L'histoire commence à Sumer*, Champs, Flammarion, 1994.

M. ASCHER, *Mathématiques venues d'ailleurs*, Seuil/sciences ouverte, 1998.

A. DABAN-DALMEDICO, J. PEIFFER, *Une histoire des mathématiques*, Coll. Points S49, Seuil, 1986.

Alain TAURISSON, *Du boulier à l'informatique*, La Cité des Sciences et de l'Industrie, Explora, Presses Pocket, 1991.

F. RAUSSER, D. BONHÔTE, F. BAUD, *Au temps des boîtes à musique*, Editions Mondo, 1972.

Jean LÉVI, *Le Grand Empereur et ses automates*, Albin Michel, 1985, Poche 6239.

Brano JACOMY, *Une histoire des techniques*, Coll. Points S67, Seuil, 1990.

Philippe BRETON, *Une histoire de l'informatique*, Coll. Points S65, Seuil, 1990.

En anglais:

M. R. WILLIAMS, *History of Computing Technology*, IEEE Computer Society Press, Los Alamitos, CA, second edition, 1997.

D. RUTLAND, *Why Computers are Computers*, Wren Publishing, 1995.

David HAREL, *Algorithimcs: The Spirit of Computing*, Addison-Wesley, 1987

Jean E. SAMMET, *Programming Languages: History and Fundamentals*, Englewood Cliffs, N.J.: Prentice-Hall. 1969.

T. J. BERGIN & R. G. GIBSON, *History of Programming Languages*, Tome I & II, Addison-Wesley Publishing Company, 1996 (ETH 768 796)

M. S. MALONE, *The Microprocessor. A Biography*, L995, Springer-Verlag, New York, Inc., published by Telos, Santa-Clara, CA.

T. R. REID, *The CHIP: How Two Americans Invented the Microchip and Launched a Revolution*, Random House Trade Paperbacks, New York, second edition, 2001.

Martin DAVIS, *The Universal Computer: The Road from Leibnitz to Turing*, W.W. Norton & Company, New York, 2000.

Petit historique des langages de programmation

1946 Konrad Zuse développe Plankalkül
1949 Short Code, premier langage de programmation utilisé
1951 Grace Hopper développe le compilateur A-O qui deviendra Math-Matic
1952 Alick Glennie développe le compilateur AutoCode
1954 IBM commence le développement de FORTRAN
1957 FORTRAN résulte des travaux dirigés par John Backus
1958 Présentation de FORTRAN II
1958 John McCarthy commence le développement de LISP
1958 Première définition d'ALGOL
1959 COBOL est créé lors de la conférence CODASYL

1960 ALGOL 60 est publié et connaît un succès surtout en Europe
1962 FORTRAN IV est lancé
1964 John Kemeny et Thomas Kurtz inventent BASIC
1967 SIMULA est créé à Oslo en Norvège
1968 Niklaus Wirth commence le développement de Pascal
1970 Développement de Smalltalk à Xerox PARC
1972 Dennis Ritchie crée le langage C
1973 Gary Kildall crée PL/M, premier langage pour microprocesseurs
1973 Alain Comerauer fait paraître la première implémentation de Prolog
1975 Dr Wong développe Tiny BASIC et Bill Gates vend son BASIC à MITS
1978 Publication du standard pour FORTRAN 77
1980 Bjarne Stroustrup développe un C avec classes, fondement de C++
1983 Publication de Smalltalk-80 par Goldberg
1983 Apparition d'Ada
1986 Apparition de C++
1991 Visual BASIC gagne le prix Best of Show de BYTE
1994 Sun annonce le langage Java

Petit historique des OS

1956 Moniteur pour l'IBM-704, General Motors, ébauche d'OS
1961 CTSS (*Central Time Share System*), MIT, premier système à temps partagé
1964 OS/360 (*Operating System*), IBM pour IBM-360, système OS batch avec multiprogrammation
1968 MULTICS (*Multiplexed Information and Computing System*), MIT, OS batch avec multiprogrammation et mémoire virtuelle
1969 UNIX, Bell Labs pour DEC PDP-7, OS avec multiprogrammation et temps partagé
1970 TOPS (*Time Shared Operating System*), DEC pour DEC-10, OS temps partagé
1974 MVS (*Multiple Virtual Storage*), IBM pour IBM-370, OS temps partagé, mémoire virtuelle
1975 UNIX, Bell Labs, premier OS écrit en langage C indépendant du processeur
1976 CP/M (*Control Program for Microcomputers*), Digital Research pour Intel 8080 et Zilog Z-80, premier OS pour microprocesseurs
1978 VMS (*Virtual Memory System*), DEC pour VAX-11
1981 MS-DOS (Disk Operating System), Microsoft pour Intel 8086, PC-DOS est la version à succès pour IBM-PC
1982 CP/M 86, Digital Research pour IBM PC
1983 Lisa (*Local Integrated Software Architecture*), Apple pour Lisa, premier OS pour PC avec fenêtrage, ballon d'essai pour MacOS
1987 Windows 2, Microsoft pour Intel 80286, représente la transition difficile de DOS vers un vrai système de fenêtrage
1987 OS/2 (*Operating System 2*), Microsoft pour IBM PS/2, l'alliance entre Microsoft et IBM sur OS/2 cesse en 1992
1991 OSF/1 (*Open Systems Foundation*), alliance de IBM, HP, DEC en compétition avec UNIX
1993 Windows/NT, Microsoft pour Intel 80?386, premier Windows a supporter l'architecture 32 bits
1994 Linux, la version 1.0 est rendue publique

INDEX

LES AUTEURS

Christian Piguet a obtenu son diplôme d'ingénieur de l'Ecole polytechnique fédérale de Lausanne en 1981 et un doctorat de cette même école en 1981. Dès 1974, il travaille au Centre Electronique Horloger S.A., à Neuchâtel, puis dès 1984, au Centre Suisse d'Electronique et de Microtechnique SA (CSEM), où il est actuellement chef du secteur «Ultra Low Power». Il travaille sur des méthodes de conception de circuits intégrés à très basse puissance. Christian Piguet est également professeur à l'EPFL, à l'Université de Lugano (ALaRI) et à l'Université de Neuchâtel, ainsi que dans de nombreux cours postgrades en Suisse et à l'étranger. Auteur d'une trentaine de brevets ainsi que d'environ 150 publications scientifiques et de quelques livres, il est membre de plusieurs comités de programmes de conférences internationales. Il a par ailleurs été président ou coprésident de comités de programme de plusieurs conférences internationales.

Heinz Hügli est directeur de recherche et professeur associé à l'Institut de microtechnique (IMT) de l'Université de Neuchâtel où il dirige le laboratoire de Reconnaissance des formes. Diplômé en électrotechnique de l'EPFZ et Dr. sc. techn. de la même école, il s'est spécialisé dans le traitement numérique des images à l'Institut de Physique Technique de l'EPFZ puis à l'Université de Californie du Sud à Los Angeles. En 1988 il crée à l'IMT le laboratoire dont la recherche actuelle porte sur la vision artificielle avec un accent en vision préattentive, microvision ainsi qu'en reconnaissance et modélisation tridimensionnelle. Heinz Hügli enseigne à l'Université de Neuchâtel dans les domaines des microprocesseurs, du traitement des images, de la perception artificielle et de la reconnaissance des formes. Auteur de plus de 120 publications scientifiques, il est membre de comités de programme de conférences internationales, de divers conseils et sociétés scientifiques et a présidé des conférences internationales.