Architecture des ordinateurs

tome 1

Le sous-système central

J.-A. MONTAGNON E. PICHAT

Préface de G. LEPICARD

MASSON

ARCHITECTURE DES ORDINATEURS

Tome 1

CHEZ LE MÊME ÉDITEUR

Dans la collection Manuels Informatiques Masson :

TECHNIQUE DE LA PROGRAMMATION, par G. HARDOUIN-MERCIER. 1974, 152 pages.

STRUCTURE DE L'ORDINATEUR. Notions de base, par R. DOWSING et F. WOODHAMS. 180 pages environ, *à paraître.*

LES SYSTÈMES D'EXPLOITATION. Structure et concepts fondamentaux, par C. LHERMITTE. 1985, 188 pages.

SYSTÈME D'EXPLOITATION. Conception et fonctionnement, par D. BARRON. 1986, 160 pages environ.

UNIX. Système et environnement, par A.-B. FONTAINE et P.H. HAMMES. 1986, 3e tirage, 232 pages.

MULTICS. Guide de l'usager, par J. BERSTEL et J.-F. PERROT. 1986, 280 pages.

LE MICROPROCESSEUR MC 68000. Contrôle et mise au point des composants associés, par J.W. COFFRON. Traduit de l'anglais par M. BOULAFA. 1985, 216 pages.

LE MICROPROCESSEUR 16 BITS-8086/8088. Matériel, logiciel, système d'exploitation, par A.-B. FONTAINE. 1984, 2e édition étendue aux 186 et 286, 240 pages.

Autres ouvrages :

COMPUTERS AND COMPUTING/INFORMATIQUE ET CALCUL. Textes publiés par P. CHENIN, C. DI CRESCENZO et F. ROBERT. Collection *Etudes et Recherches en Informatique.* 1986, 408 pages.

LOGIQUE BINAIRE. Fonctions logiques et arithmétique binaire, par M. AUMIAUX. 1984, 2e édition, 4e tirage, 336 pages.

MICROPROCESSEURS 8 BITS, par M. AUMIAUX. 1985, 4e édition, 2e tirage, 320 pages.

MICROPROCESSEURS 16 BITS, par M. AUMIAUX. 1985, 224 pages.

LES SYSTÈMES À MICROPROCESSEUR, par M. AUMIAUX. 1982, 2e édition révisée et augmentée, 272 pages.

ARCHITECTURE DES ORDINATEURS

Tome 1

LE SOUS-SYSTÈME CENTRAL

J.-A. MONTAGNON

Ingénieur ESE
Docteur ingénieur IMAG
Consultant en réseaux d'entreprise

E. PICHAT

Ingénieur des Arts et Manufactures
Docteur ès Sciences
Professeur au Conservatoire National des Arts et Métiers

Préface de G. LEPICARD

MASSON
Paris New York Barcelone Milan
Mexico São Paulo
1986

Illustration de la couverture :
Labyrinthe de la Villa Pisani à Stra.
D.R.

ISBN : 2-225-80915-1

MASSON S.A.	120, bd Saint-Germain, 75280 Paris Cedex 06
MASSON PUBLISHING U.S.A. Inc.	1 Ames Court, Plainview, N.Y. 11803
MASSON S.A.	Balmes 151, 08008 Barcelona
MASSON ITALIA EDITORI S.p.A.	Via Giovanni Pascoli 55, 20133 Milano
MASSON EDITORES	Dakota 383, Colonia Napoles, 03810 Mexico D.F.
EDITORA MASSON DO BRASIL Ltda	Rua Borges Lagoa 1044, CEP 04038 São Paulo S.P.

préface

Ces tomes décrivent l'architecture des systèmes informatiques sur les plans du matériel et des systèmes d'exploitation.

Les auteurs, Jean-Antoine Montagnon et Etienne Pichat, possèdent une solide expérience de la conception et de l'utilisation des systèmes informatiques, ainsi que de leur enseignement. Jean-Antoine Montagnon a participé dans la Compagnie des Machines Bull à la conception de la ligne de calculateurs 64 et DPS7 qui a connu un succès mondial. Le constructeur japonais NEC en a acquis la licence pour développer sa ligne de produits ACOS 4. Etienne Pichat a une longue carrière universitaire, consacrée à la recherche et à l'enseignement de l'informatique.

Ce Tome I décrit l'architecture matérielle des systèmes centraux. Le thème important des interfaces entre le matériel et le logiciel est largement couvert avec une présentation des différentes solutions, de leurs avantages et de leurs inconvénients.

Les systèmes d'exploitation, les appareils périphériques sont décrits dans les autres tomes.

Cet ouvrage présente une synthèse des typologies des différentes architectures des systèmes informatiques et les orientations des architectures nouvelles.

Cet ouvrage intéressera les informaticiens avertis qui désirent approfondir leurs connaissances des systèmes informatiques. Il sera également utile aux étudiants en informatique. Il est aussi conseillé aux personnes qui désirent acquérir une connaissance des systèmes informatiques.

Georges Lepicard.

Directeur chez Bull, M. G. Lepicard fut responsable de la conception des 64 et DPS7.

table des matières

introduction

"Il faut toutefois préserver l'optimisme un peu candide, sans lequel ces sortes de synthèse ne seraient jamais entreprises."
E. Berl.

Ce tome est le premier d'une série décrivant l'architecture des systèmes informatiques,

- tant matérielle que logicielle,
- qu'elle soit centralisée ou répartie.

Ce premier tome est plus particulièrement consacré à ce que nous définirons comme étant le **"sous-système central"**. Les tomes suivants le compléteront avec le découpage suivant:

- tome II : **systèmes d'exploitation** et extensions du matériel associées. On décrira le rôle, la typologie, la structure des systèmes d'exploitation. Sur le plan du matériel nous introduirons: pagination, segmentation et capacités;

- tome III : **entrées-sorties**: description des processeurs d'entrée-sortie et canaux, des mécanismes d'entrée/sortie et d'interruption. On y trouvera aussi un chapitre détaillé sur les bus;

- tome IV : **périphériques**: y seront décrits les sous-systèmes périphériques, et tout particulièrement ceux qui assurent la fonction de stockage comme les bandes et les disques. Les écrans et imprimantes, appareils les plus visibles des utilisateurs, y figureront aussi;

- tome V : **développement de programme**, c'est-à-dire l'ensemble des transformations que subit un programme écrit dans un langage de haut niveau pour obtenir un programme en langage machine exécutable par le matériel de l'ordinateur. Les sujets abordés seront la compila-

tion, l'assemblage, l'édition de liens, le chargement, la structure et l'utilisation des sous-programmes, les bibliothèques.

Le domaine traité est très vaste. Les différents tomes dégagent les concepts utilisés pour les ordinateurs et les illustrent de façon aussi complète que possible. Ils intéressent:
- les enseignants et les étudiants,
- les professionnels de l'informatique voulant élargir ou approfondir leur domaine de compétence,
- les esprits curieux qui souhaitent comprendre les ordinateurs à l'aide d'une approche structurée et mieux les utiliser,
- tous ceux qui veulent suivre l'évolution des systèmes informatiques.

Ces différents tomes sont le fruit d'une longue expérience professionnelle et d'une réflexion poursuivie au contact de très nombreux publics: formation continue au Conservatoire National des Arts et Métiers (CNAM) ou promotion supérieure du travail (CESI), aux Ponts et Chaussées, formation du personnel de l'Education Nationale, formation de base dans des écoles d'ingénieurs comme l'Institut d'Informatique d'Entreprise (IIE), dans des Universités (Maîtrise d'informatique, MIAGE) et écoles supérieures de commerce (CERIX), dans des entreprises. Le contenu de ce tome a été enseigné par les auteurs aux différentes catégories citées ci-dessus, depuis une dizaine d'années. Ils utilisent une méthode qui n'a cessé d'être remise en cause et améliorée chaque année afin d'offrir une approche claire, progressive et complète. Ils ne cessent de mettre à jour régulièrement la matière de l'ouvrage.

Sur le plan du matériel, ces ouvrages seront illustrés par des microprocesseurs (6502, 8086, etc.), mais aussi des minis et des très gros ordinateurs (DPS88, 308X, etc.), par des bus SASI, des canaux. Pour le logiciel ils feront aussi bien référence à CP/M qu'à MS-DOS, UNIX, GCOS de Bull, ou MVS d'IBM.

Les "couches basses" du matériel comme l'électronique, la technologie des composants, le conditionnement (packaging), le détail des circuits ne sont cependant pas approfondis.

Certains chapitres se situent aux confins du sujet étudié. Leur lecture n'est pas nécessaire a priori pour le lecteur initié, elle permettra cependant au lecteur désireux de préciser ses acquis et de faire le lien avec d'autres domaines plus techniques ou plus théoriques, d'acquérir une terminologie, un résumé de l'état de l'art et des indications bibliographiques.

Ce tome lui-même n'échappe d'ailleurs pas à cette règle et certains lecteurs familiarisés avec les sous-systèmes centraux pourront très bien passer directement au Tome II.

Pour illustrer les notions présentées au cours des différents chapitres, nous utiliserons un des microprocesseurs les plus répandus, le 8086 d'INTEL, que l'on trouve sur la majorité des micro-ordinateurs, en particulier sur les compatibles du PC d'IBM. Pour des

raisons pédagogiques, nous utiliserons cependant souvent le 6502 que l'on trouve sur de nombreux ordinateurs familiaux (Apple II, Pet de Commodore, Atom d'Acorn, Oric, etc.) et même sur des commutateurs de paquets (TP4000 de GTE-TELENET) dans le domaine des réseaux. Le 6502 est le microprocesseur huit bits le plus vendu après le Z80. Il a l'avantage d'être plus simple que le Z80 tout en disposant des fonctionnalités de base.

Dans le domaine des ordinateurs de gestion plus gros nous prendrons de préférence des exemples chez IBM, le constructeur dominant, ainsi que chez Bull, le constructeur national. Mais des machines aussi répandues que les VAX de DEC ne seront pas ignorées.

En ce qui concerne les langages nous ferons appel au langage machine, au langage d'assemblage lorsque nécessaire, nous emploierons le Basic et le C dans les autres cas.

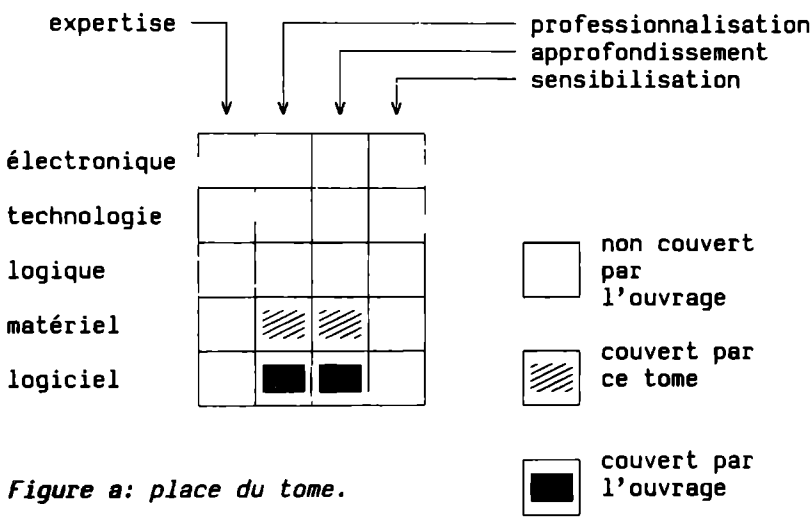


Figure a: place du tome.

Nous avons préféré un tome avec de nombreux chapitres, courts pour la plupart, qu'avec quelques chapitres longs et un découpage fastidieux en sous-chapitres de huitième niveau. Les sujets qu'ils recouvrent sont suffisamment bien délimités pour qu'ils aient chacun leur propre unité et leur propre équilibre.

Les chapitres qui le nécessitent font l'objet d'un résumé rappelant les notions principales qui ont été introduites.

Ce tome est écrit en français, et l'on constatera qu'il est effectivement possible de parler d'ordinateur en français, avec un langage parfois moins ambigu que les termes anglo-saxons qui ne sont pas toujours très bien compris par ceux qui les emploient. L'impor-

tance de l'américain est cependant telle dans le domaine des ordinateurs que nous donnerons bien évidemment la correspondance entre termes français et américains.

Les références bibliographiques figureront dans le texte sous forme de crochets contenant:

- les trois premières lettres du nom du premier auteur,
- les deux chiffres du millésime,
- une lettre éventuelle pour distinguer les doublons.

Les tableaux et figures sont repérés par le numéro du chapitre courant, suivi d'une ou de deux lettres (aa vient après z) minuscules.

oOo

Nous tenons à remercier particulièrement M. Pierre Marquès pour les conseils et l'aide essentiels qu'il nous a apportés tout au long de la réalisation de ce tome, ainsi que le lieutenant-colonel Denis Fargette, MM. Alain Autin, de Closmadeuc, Gilles Guerrin, Pierre Montagnon, Alain Pinet, Christian Pouly qui ont bien voulu relire tout ou partie de ce tome et nous faire bénéficier de leurs remarques. Nous les en remercions vivement.

chapitre 1

définition de l'architecture des ordinateurs

1.1 – Ce que l'on entend en général par architecture.

Si le terme d'architecture est précisément défini et compris dans certains domaines comme le bâtiment, il n'en est pas de même en informatique, et tout particulièrement dans le domaine du matériel. Nous prendrons l'exemple de la présentation de l'architecture de leurs ordinateurs faite par quelques constructeurs: cela va de "l'architecture de base des DPS88" de Bull (figure 1a) dont le commentaire du constructeur indique que "ce schéma donne une représentation fonctionnelle du DPS88" à l'architecture de l'UC selon [MIT84].

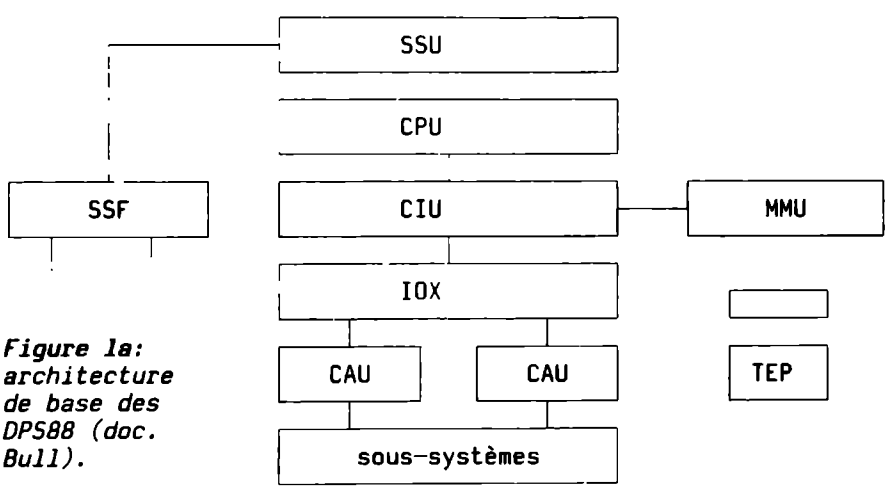


Figure 1a: architecture de base des DPS88 (doc. Bull).

Ces exemples illustrent les acceptions diverses du terme d'architecture:
- diagramme fonctionnel,
- configuration du matériel,

selon le niveau où l'on se place.

Pour souligner la difficulté d'expliquer ce que **tous** les intéressés entendent par architecture dans le domaine des ordinateurs, nous citerons quelques définitions, faites par différents auteurs:
- J.L. Baer auteur d'un livre sur l'architecture des ordinateurs,
- INTEL fabricant de composants et d'ordinateurs basés sur ses composants,
- F. Brooks chef de projet de l'IBM 360,
- DEC constructeur d'ordinateurs.

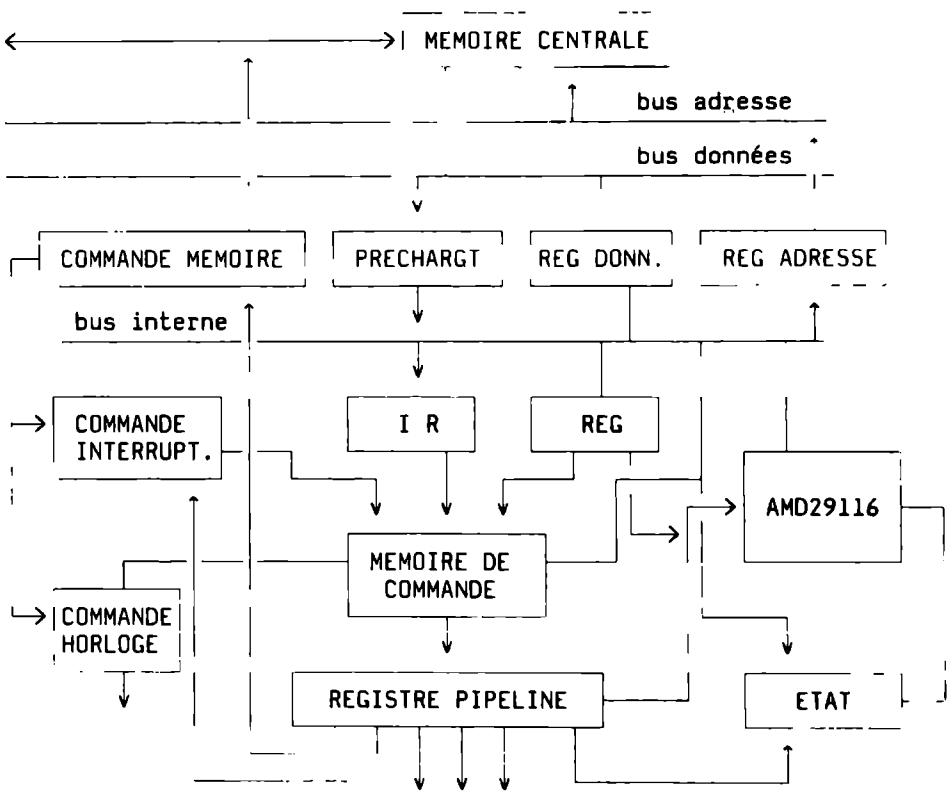


Figure 1b: architecture de l'unité centrale selon [MIT84].

1.1.1 – Définitions de J.L. Baer.

Baer propose cinq niveaux d'architecture résumés ci-dessous:

- niveau 1 : il correspond à la structure globale du système (processeurs, canaux, modules de mémoire, périphériques d'entrée/sortie avec leurs interconnexions),
- niveau 2 : à ce niveau on trouve la description du processeur (jeux d'instructions, représentation interne des données, déroulement des instructions),
- niveau 3 : transferts entre registres et une unité fonctionnelle,

- niveaux 4 et 5 : conception logique et circuits (portes, bascules, transistors, etc.).

Baer propose aussi une notation (appelée PMS) des différentes classes d'éléments que l'on trouve dans l'ordinateur:
L lien entre composants transférant l'information sans l'altérer,
M stocke l'information sans la modifier,
S représente un commutateur,
K représente une unité de commande,
D représente un opérateur,
T est un convertisseur (transducer) permettant de communiquer avec l'extérieur, il change la représentation de l'information sans en changer le sens (la signification),
P est un processeur, c'est-à-dire un ensemble de M+K+D+L+S, un cas particulier en somme.

1.1.2 – Définition d'INTEL.

On trouve dans [INT81]: "Le terme «architecture d'ordinateur» est souvent utilisé comme signifiant simplement «organisation et conception des ordinateurs». En pratique il y a plusieurs architectures distinctes dans un système informatique, chacune définie par une frontière entre différents niveaux du système.

Une architecture est la frontière entre deux des modules (ou briques) fonctionnels ci-dessus. Elle pourraît être définie comme «l'apparence fonctionnelle du système en-dessous d'une interface, pour un utilisateur placé au-dessus de cette interface». Les concepteurs se plaçant au-dessus d'une interface architecturale ne seront généralement pas concernés par les détails du système sous-jacent ou par des architectures de plus bas niveau; ils ne s'intéressent qu'à la fonction du système offerte à l'interface immédiatement en-dessous d'eux."

1.1.3 – Définition de F. Brooks.

Dans son article "The mythical man-month", F. Brooks résume l'architecture comme étant "la spécification complète et détaillée de l'interface utilisateur".

En restant dans le monde IBM, on peut lire page 1-1 des spécifications de l'architecture XA ([IBM83F]): "L'architecture d'un système définit ses attributs tels que les voit le programmeur, c'est-à-dire sa structure conceptuelle et le comportement conceptuel de la machine; elle ne s'intéresse pas par contre à l'organisation du flot des données, à la conception logique, à la conception physique, aux performances d'une implémentation donnée".

1.1.4 – Définition de DEC.

Dans le "VAX-11 Architecture Handbook" on trouve page 1: "Un objectif initial important a été de s'assurer que tous les programmes VAX s'exécutent sur chaque membre de la famille VAX-11. L'ensemble des attributs que tous les membres de la famille ont en commun et qui assurent la compatibilité logicielle est collectivement appelé l'architecture VAX".

1.1.5 – Autres définitions.

[GIL82] dit que "l'ordinateur numérique effectue les calculs sous le contrôle d'un **programme**. L'organisation générale selon laquelle s'effectuent ces calculs constitue ce que l'on appelle **l'architecture** de l'ordinateur."

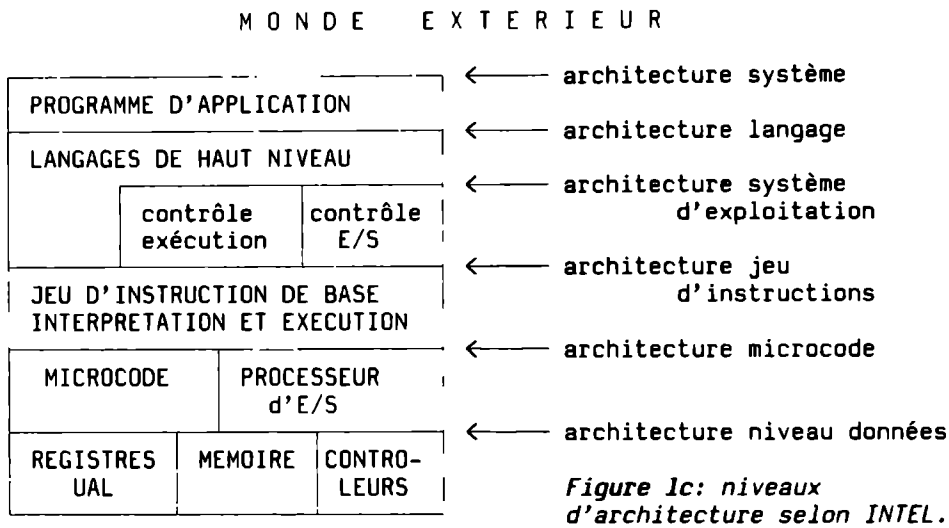


Figure 1c: niveaux d'architecture selon INTEL.

1.2 – Ce que nous retiendrons dans ce tome.

Nous avons vu ci-dessus combien il est difficile de s'aligner sur une autorité extérieure et suffisante pour statuer sur la définition de l'architecture des ordinateurs. Afin de donner une cohérence et une clarté dans l'exposé de ce tome, nous définirons des architectures du **matériel** à trois niveaux différents:
- l'architecture externe,
- l'architecture interne,
- la structure physique.

Cette approche utilisée dans nos enseignements permet de mieux classer tous les objets que l'on trouve dans un ordinateur, de les introduire progressivement, et toute approche classificatrice est propice à une structuration, donc à une meilleure compréhension par des non initiés. Elle correspond de plus à une réalité que nous avons pu connaître en tant que collaborateur de grands constructeurs, mais aussi en tant qu'utilisateur.

1.2.1 – Architecture externe.

L'architecture externe (ou fonctionnelle) correspond au niveau d'abstraction le plus élevé, en gros à la limite (l'interface dans la terminologie informatique) entre le matériel et le logiciel pour ce

qui concerne le sous-système central. Elle correspond au niveau 2 de J.L. Baer, à "l'architecture du jeu d'instructions" de [INT81], à "l'interface utilisateur" décrite dans les "Principles of Operation" d'IBM chez F. Brooks.

L'architecture externe est encore l'ensemble des règles que doivent vérifier plusieurs ordinateurs s'ils veulent être "compatibles", et en particulier appartenir à la même "gamme":
- les différents modèles de la gamme 360 d'IBM suivent le "Principles of Operation" de cette gamme,
- les différents modèles de la gamme 370 ont eu aussi leur propre "Principles of Operation",
- de même pour la gamme 43XX,
- les DPS7 de Bull,
- les DPS8 de Bull de leur côté,
- les VAX de DEC,
- etc.

Nous définirons plus précisément cette architecture externe dans le chapitre 2.

On remarquera dans les exemples ci-dessus que les architectures externes évoluent (beaucoup plus lentement que les architectures internes), les nouvelles étant compatibles avec les précédentes dans le sens ascendant, comme l'illustrent les figures 1d à 1f.

Définir une architecture externe est une décision majeure pour un constructeur, la décision finale est une synthèse de points de vue technique, technologique et surtout **économique**.

C'est l'architecture externe qui fait que des programmes vont continuer à fonctionner. Elle constitue ainsi une frontière économique, en assurant la préservation de l'investissement.

1.2.2 – Architecture interne.

A la différence de l'architecture externe, l'architecture interne a une portée moins générale: plusieurs ordinateurs (plusieurs modèles) relevant de la même architecture externe, donc appartenant à une même gamme, auront par contre des architectures internes différentes. Ainsi
- dans le modèle haut de gamme les opérateurs travailleront sur 32 bits simultanément (en parallèle, cas des 68020 de Motorola) alors qu'en bas de gamme ils ne travailleront que sur 16 bits en parallèle (cas du 68000 chez Motorola),
- dans un modèle plus puissant, les données seront échangées sur 16 bits en parallèle entre la mémoire centrale et le processeur central (cas du 8086 d'INTEL) alors qu'un modèle moins puissant ne disposera que de 8 bits pour ses échanges (cas du 8088),
- dans un mini-ordinateur, le processeur central disposera de vrais registres alors qu'ils seront simulés en mémoire centrale dans le cas d'un microprocesseur relevant de la même architecture externe (cas de Texas Instruments),
- etc, le nombre d'exemples analogues peut être multiplié facilement.

L'architecture interne, contrairement à l'architecture externe, n'est pas vue par le logiciel et n'a donc pas besoin d'être comprise

par un programmeur ou un concepteur de logiciel travaillant au niveau de ce que l'on appelle le langage d'assemblage. Elle ne concerne pas seulement les objets logiques comme les données, les instructions, les interruptions, etc, mais aussi des objets physiques qui constituent matériellement l'ordinateur: la façon dont les registres sont physiquement réalisés et reliés, les bus, les opérateurs, les cycles, les phases, etc.

Si le "VAX-11 Architecture Handbook" correspond à notre architecture externe, le "VAX Hardware Handbook" correspond à l'architecture interne, avec un chapitre pour chaque modèle.

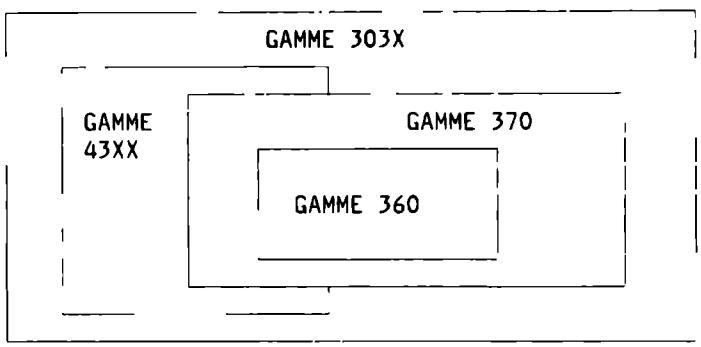


Figure 1d: gammes IBM issues des 360.

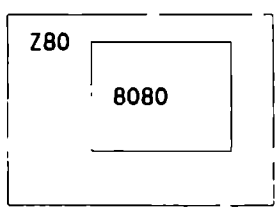


Figure 1e: gammes ZILOG issues des 8080.

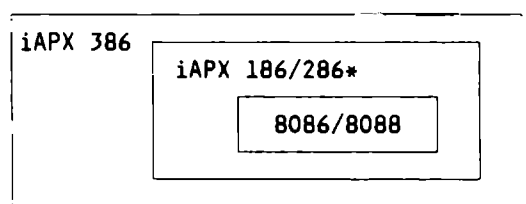


Figure 1f: gammes d'INTEL issues des 8086. (: cf. [CLA85])*

1.2.3 – Structure physique.

La structure physique est celle qui est la plus visible de l'extérieur, c'est-à-dire celle que voit le simple visiteur à qui l'on ouvre la porte des armoires, alors que l'architecture externe est celle que voit le programmeur travaillant au niveau du langage machine. Elle se présente sous forme d'armoires, de baies, de clés, de voyants, d'unités de refroidissement, de ventilation, d'alimentation, etc. Elle ne jouera dans cet ouvrage qu'un rôle mineur par rapport aux deux précédentes.

1.3 – Remarques sur le terme d'architecture.

Etant donné la grande diversité des membres de notre comité de lecture qui comprenait des néophytes, des étudiants, mais aussi des

experts en architecture en activité chez de grands constructeurs, ce découpage en trois a suscité des discussions vives et enrichissantes.

Un premier point discuté concerne le terme d'architecture. Le Larousse le définit comme l'art de construire, de disposer, d'orner les édifices, en signalant qu'au figuré il est synonyme de forme, de structure. Ce tome n'est pas un livre sur l'art de construire, de disposer ou orner les ordinateurs. Il est par contre un ouvrage sur leur structure.

Un second point concerne la décomposition, le partitionnement en couches très évident dans le schéma de [INT81]. Cette approche en couches (layer) popularisée ces derniers temps par le modèle de l'ISO dans le domaine des réseaux était souvent appelée structure en pelures d'oignon auparavant lorsque l'on parlait de logiciel.

Définir les couches ainsi que leurs interactions est éminemment un travail d'architecte. Ce n'est cependant pas une panacée dans la mesure où les couches conduisent à des résultats lourds qui pénalisent le matériel. Elle est malgré tout inéluctable quand on considère l'ampleur des problèmes posés et elle reflète la structure organisationnelle mise en place pour résoudre les grands problèmes.

Le rôle de l'architecte peut alors apparaître comme un rôle **vertical**, transversal: il a une certaine connaissance d'ensemble des couches et fera en sorte que leur fonctionnement global ne soit pas excessivement lourd dans les cas les plus fréquents de fonctionnement. Il transcende les interfaces. Il y a en particulier des interactions continues entre le logiciel et le matériel: les nouvelles gammes, compatibles avec les précédentes de façon ascendante, intègrent dans le matériel de nouvelles fonctions qui améliorent l'ensemble matériel-logiciel. La microprogrammation que nous verrons au chapitre 11 a beaucoup facilité cette évolution. Un exemple en est constitué par les aides microprogrammées (assist) que l'on trouve dans les processeurs d'IBM.

L'architecte utilisera pour cela des méthodes de calcul basées sur les files d'attente, tout comme la résistance des matériaux est l'outil de base dans le calcul des structures des édifices. Le lecteur intéressé par cette discussion trouvera des éléments supplémentaires dans [BIR85].

1.4 – Résumé.

ARCHITECTURE EXTERNE	ARCHITECTURE INTERNE
. données	. bus
. instructions	. opérateurs
. entrées/sorties	. mémoires
. interruptions	. phases
. registres/mémoire centrale	. cycles
	. antémémoires
	. registres de traduction
. etc.	. etc.

chapitre 2

l'architecture du sous-système central

2.1 – Notion de sous-système.

Comme tout objet, tout système complexe dont la description requiert une décomposition, un ordinateur peut être en première analyse décrit comme un ensemble de sous-systèmes complémentaires dont seule la réunion offre une fonction significative pour l'utilisateur.

Dans cette première analyse, volontairement simplifiée, mais non pas simpliste, nous aborderons l'ordinateur comme la réunion d'un sous-système central et de sous-systèmes périphériques, conformément au schéma:

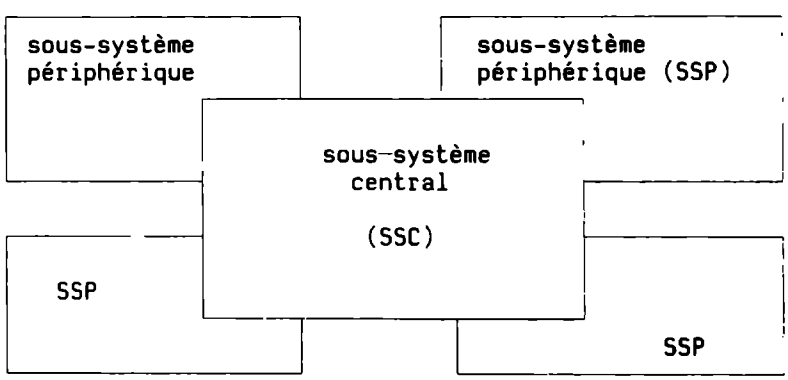


Figure 2a: décomposition de l'ordinateur en sous-systèmes.

Chacun de ces sous-systèmes a des fonctions clairement définies, et le sous-système central est le seul à communiquer directement avec les sous-systèmes périphériques qui ne peuvent dialoguer entre eux. Dans certains ordinateurs l'interface entre le sous-système central (SSC) et les sous-systèmes périphériques (SSP) est totalement banalisée:

- quelque soit le type de sous-système périphérique considéré (disque, imprimante, unité de communication, etc.),
- au niveau des actions lancées par le sous-système central (programmes canal des 370 ou DPS7), niveau relèvant de l'architecture externe,
- au niveau des câbles et du dialogue entre le sous-système central et les sous-systèmes périphériques (canal multiplexeur par blocs par exemple), niveau qui relève de l'architecture interne.

2.2 – Définition du sous-système central.

Lorsque l'on se réfère à la définition de l'informatique et d'un ordinateur on constate que ce dernier doit assurer des fonctions de:
- traitement de l'information,
- stockage de l'information,
- communication avec l'extérieur de l'ordinateur.

Si les deux dernières fonctions sont assurées principalement par les sous-systèmes périphériques, la fonction de **traitement** l'est par le sous-système central.

Pour assurer cette fonction, le sous-système central dispose de ses ressources internes (mémoires, opérateurs, chemins que suivent les données), mais de plus il **pilote** les ressources des sous-systèmes périphériques (disques, bandes, imprimantes, etc.) qui ne font preuve d'aucune initiative: seul le sous-système central prend des décisions qui soit conduisent à des actions qui lui sont purement internes, soit sollicitent l'aide des sous-systèmes périphériques.

Le sous-système central assure donc un double rôle:
- il assure la fonction de traitement,
- il pilote l'ensemble de l'ordinateur.

2.3 – Traitement de l'information.

Sans entrer dans de subtiles différenciations entre données, informations, connaissances, qui n'ont pas d'incidence au niveau d'abstraction où nous nous plaçons, il faut préciser ce que nous entendons par "traitement" de l'information.

Aujourd'hui, nombreux sont ceux qui voient dans l'ordinateur une super-machine à calculer, mais la réalité est plus complexe. Faire des calculs correspond bien à traiter de l'information au sens où nous l'entendons, cependant:
- les nombres ne sont pas les seules **informations** traitées par l'ordinateur. Il sait aussi traiter des noms, rechercher des mots, des chaînes de caractères plus généralement;
- les calculs ne sont pas les seules **opérations** que sait exécuter l'ordinateur. Il sait aussi faire des comparaisons et dérouler des traitements différents selon les résultats de ces comparaisons. Il prend des "décisions".

Ces décisions qu'il peut prendre à la suite de comparaisons, et surtout **l'enchaînement automatique** des opérations, sans intervention humaine entre chaque opération comme dans le cas de la machine à cal-

culer, sont les caractéristiques les plus significatives de l'ordinateur lorsqu'on veut le comparer à la machine à calculer.

Sa **vitesse de traitement**, incomparablement plus grande que celle obtenue avec une machine à calculer, n'est qu'une conséquence des caractéristiques précédentes qui peuvent bénéficier automatiquement d'une technologie de rapidité toujours croissante: les plus gros ordinateurs spécialisés dans le calcul atteignent ainsi des vitesses de plus de 1 milliard de résultats scalaires (nombres flottants calculés) par seconde en 1985/86

Alors qu'avec une machine à calculer l'utilisateur réalise de façon répétitive le cycle d'actions ci-dessous:

1. frappe d'un nombre,
2. frappe du symbole de l'opération (en excluant le cas des machines basées sur la notation polonaise),

les opérations que doit enchaîner automatiquement l'ordinateur sont **toutes** lues par l'ordinateur et rangées dans des mémoires (rôle de la mémoire centrale) avant d'être exécutées automatiquement (rôle du processeur central). Les opérations ainsi rangées dans l'ordinateur préalablement à leur exécution forment ce que nous appelons un **programme**.

Certains lecteurs vont alors se demander pourquoi nous n'avons pas utilisé le terme très répandu **d'unité centrale** que l'on retrouve dans les acronymes UC (CPU pour Central Processing Unit en américain, fréquemment associé au temps dans l'expression "temps CPU") et qui correspond bien à ce que nous appelons sous-système central, encore que CPU ne concerne que le processeur central en américain, et qu'UC comprend la mémoire centrale pour certains en français. Notre attitude est basée sur le fait que le terme d'unité centrale est tombé en désuétude, tout particulièrement chez les constructeurs, là où l'on se préoccupe principalement d'architecture. Le terme "d'unité" a par ailleurs une signification très diverse: unité de disque, unité arithmétique, unité instruction, etc. L'évolution technologique n'a pas respecté ce terme qui recouvre aujourd'hui des choses très diverses en taille et en importance.

2.4 – Objets du sous-système central.

Au niveau de l'architecture externe où nous nous plaçons ici, les objets à étudier dans le sous-système central sont:

- la forme dans laquelle les opérations (les programmes) sont rangées à l'intérieur de l'ordinateur, d'où l'étude des **instructions** que nous aborderons au chapitre 4,
- les techniques selon lesquelles ces instructions nomment ou adressent les informations dans les différentes mémoires (centrales et périphériques) et communiquent avec les sous-systèmes périphériques, d'où l'étude:
  * des **modes d'adressage** au chapitre 7,
  * des **entrées/sorties** dans le Tome III,

  * des interruptions dans le Tome III: bien qu'utilisées aussi de façon purement interne au sous-système central dans les ordinateurs modernes, l'origine et la bonne compréhension du mécanisme d'interruption sont liées aux entrées/sorties;
- les types de données et la façon dont elles sont représentées dans le sous-système central, ce que nous étudierons dans les chapitres 3 et 6.

2.5 – Résumé.

Le sous-système central comprend donc (cf. figure 2b):

- le processeur central (PC) assurant le traitement,
- la mémoire centrale (MC) comme ressource interne,
- des organes de liaison et de commande (canal, processeur d'entrées-sorties, unité d'échange) pour piloter les sous-systèmes périphériques (SSP).

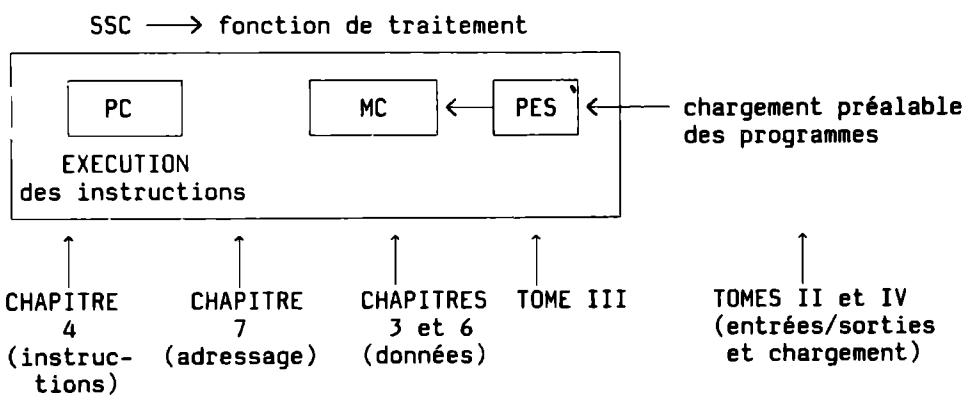


Figure 2b: les principaux chapitres.

chapitre 3

rappels sur les mémoires et le stockage des données

Ce chapitre fait partie de ceux dont la lecture est facultative parce que les notions abordées sont supposées connues du lecteur. Nous avons cependant cru utile d'en donner un résumé pour rafraîchir la mémoire des uns, pour éviter à d'autres de se plonger dès maintenant dans des ouvrages aussi copieux que celui-ci, perdant ainsi le fil du propos initial.

3.1 – Les mémoires du sous-système central.

Une mémoire est un dispositif matériel capable de conserver et de restituer l'information stockée, durant une certaine période de temps. Les mémoires les plus connues du grand public sont:
- les livres qui reposent sur la technologie du papier,
- les bandes, les cassettes audio qui utilisent le magnétisme,
- les disques microsillons qui utilisent un stockage mécanique de l'information,
- les films qui utilisent la transparence du support.

D'autres mémoires atteignent maintenant le grand public:
- les cassettes audio-visuelles des magnétoscopes,
- les disques optiques basés sur la technologie du laser.

3.1.1 – Le transistor.

Les mémoires qui nous intéressent ici utilisent la technologie des semi-conducteurs dont la matière de base dominante est encore le silicium et dont l'élément actif est le transistor.

Le transistor est, d'un point de vue général, utilisé de deux façons différentes:
- **analogique** : c'est le domaine classique de l'amplification linéaire où le signal de sortie est le signal d'entrée amplifié par un facteur appelé le gain du transistor,
- en tant qu'**interrupteur**, ou commutateur ou "en tout ou rien": le transistor est soit saturé (ou passant, la tension du collecteur est alors proche de celle de la masse), soit bloquant (la tension du collecteur est alors proche de celle de l'alimentation).

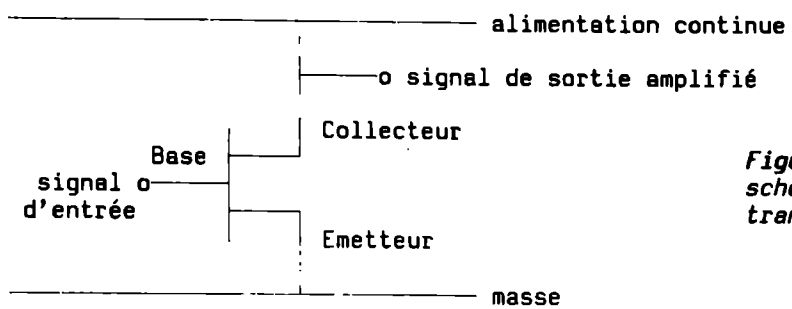


Figure 3a: schéma d'un transistor.

C'est dans son utilisation comme interrupteur que le transistor nous intéresse dans les mémoires du sous-système central: ses deux états stables saturé et bloquant (ouvert et fermé) permettent de représenter une **information binaire**, c'est-à-dire une information qui ne peut prendre que deux valeurs appelées:

- vrai ou faux,
- un ou zéro,
- etc.

Peu importe le nom que l'on donne à ces deux valeurs. Cette information binaire porte un nom particulier dans le domaine des ordinateurs, on dira que c'est un **bit** (abréviation pour "BInary digIT" ou chiffre binaire).

3.1.2 – Technologies.

Ces mémoires relèvent de deux grandes technologies:

- celle des transistors **bipolaires**, les plus rapides, que l'on retrouve avec plusieurs variantes dans les circuits les plus performants,
- celle des transistors **MOS** (Metal Oxyde Semi-conductor) moins rapides mais moins chers, plus denses, c'est-à-dire offrant une intégration plus élevée (un plus grand nombre de transistors par unité de surface de silicium). La technologie MOS a donné lieu à un grand nombre de variantes comme le CMOS (Complementary MOS) qui a l'avantage de consommer peu d'énergie.

3.1.3 – Mémoires mortes et mémoires vives.

Les fonctions que savent réaliser les mémoires à semi-conducteurs sont:

- **la lecture** : la mémoire qui reçoit un ordre de lecture fournit au demandeur, au bout d'un certain temps appelé **temps d'accès**, le contenu (l'état en fait) du transistor (ou des transistors selon le cas) qui contient l'information demandée;
- **l'écriture** : la mémoire qui reçoit un ordre d'écriture range dans le transistor (ou les transistors) désigné l'information qui lui est fournie avec l'ordre d'écriture;

- le **rafraîchissement** : cette fonction n'est nécessaire qu'avec les mémoires dynamiques (de type MOS) puisque dans ces mémoires l'information est matérialisée par une charge capacitive contenue dans le transistor. Or une capacité se décharge et il est donc nécessaire de la recharger régulièrement, toutes les 2 millisecondes par exemple.

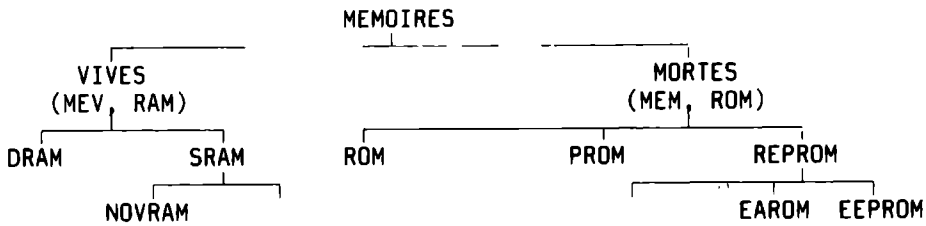


Figure 3b: types de mémoire rencontrés dans le sous-système central.

Le terme de **cycle** mémoire correspond à la fréquence maximale à laquelle la mémoire traite les requêtes qui lui sont soumises. Toutes les mémoires n'offrent pas ces fonctions:

- les **mémoires vives** (abrégé en MEV) offrent la lecture et l'écriture, mais nécessitent un rafraîchissement dans le cas des mémoires dynamiques. Le terme américain correspondant à MEV est RAM qui signifie "mémoire à accès aléatoire" (Random Access Memory). Ce terme exprime une autre caractéristique des mémoires que nous verrons plus loin;
- les **mémoires mortes** (abrégé en MEM, ROM en américain, Read Only Memory pour "mémoire à lecture seulement") n'offrent que la lecture. On ne peut donc pas y écrire, du moins simplement. En effet certaines ne sont pas définitivement figées. On trouve:
  * les mémoires programmables ou **PROM**. Leur contenu n'est pas figé à la sortie de la chaîne de production; chaque utilisateur peut au moyen d'un équipement adapté y mettre le contenu qui lui convient, c'est-à-dire la "programmer". Une fois cette première écriture effectuée on ne pourra plus en modifier le contenu;
  * les mémoires reprogrammables ou **REPROM** dans lesquelles il est possible de modifier plusieurs fois le contenu. Différents procédés sont utilisés et donnent leur nom à ces mémoires: on trouvera ainsi des EAPROM (Electronically Alterable PROM).

3.1.4 – Adressage.

Chaque information stockée dans une mémoire a une **valeur** représentée par l'état des transistors. Une information binaire (un bit) est ainsi symbolisée par 0 ou 1 (vrai ou faux).

Chaque information doit pouvoir être distinguée des autres informations stockées dans la mémoire, y être repérée. Ce repérage ne peut être réalisé en se basant sur la valeur (nous ignorerons pour l'instant les mémoires dites "associatives"), il est effectué en utilisant **l'adresse** de l'information.

Qu'est-ce qu'une adresse? Tout comme dans une rue, l'adresse d'une maison n'a aucun rapport avec le nom des personnes qui y logent,

l'adresse d'une information n'est en aucune façon reliée à sa valeur. Elle est purement arbitraire.

Une mémoire est organisée en cases, tout comme une rue est organisée en maisons, immeubles, etc. Chaque case a une adresse définie par un entier naturel qui va de zéro à N, N étant généralement une puissance de deux (deux à la puissance seize, soit 64 kilo-octets abrégé en 64 Ko sur un 6502, un Z80, le kilo K valant 1024, 2^{10}, dans le cas des ordinateurs). Chaque case comprend un certain nombre de transistors, c'est-à-dire un certain nombre de bits (8 sur un 6502), toujours le même, quelque soit l'adresse de la case. Si nous reprenons l'analogie avec la rue et les maisons, il faudra une rue avec des maisons toutes identiques.

Une information utilisera une ou plusieurs cases successives comme le montre la figure 3c.

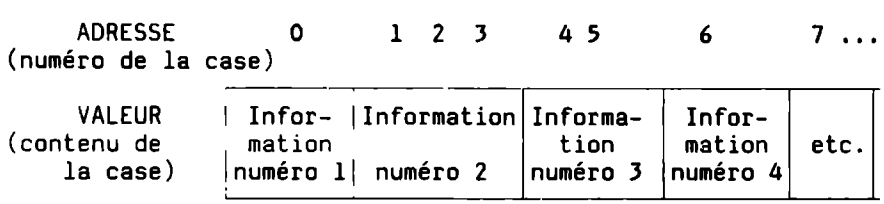


Figure 3c: relations entre adresse, case et information.

En général, nous ne manipulons pas les informations par des numéros, mais par un nom qui leur est associé. Supposons que les différentes informations présentées dans la figure 3c correspondent à différentes caractéristiques d'un individu: son âge, sa profession, son prénom, le département où il réside, etc. Nous aboutissons alors au schéma de la mémoire donné en figure 3d où l'on voit apparaître les noms des informations (on parlera de nom de "variable" dans les langages de programmation) et les adresses auxquelles elles sont rangées.

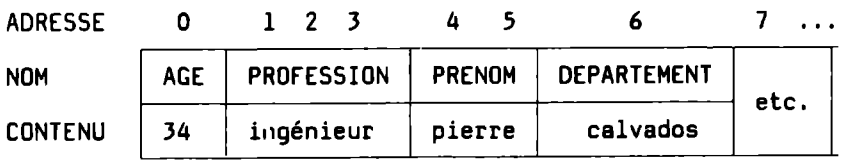


Figure 3d: relations entre case, adresse et nom d'une information.

Le choix de l'adresse est, rappelons-le, purement arbitraire: on peut choisir l'adresse 7 pour ranger l'information "AGE" sans que rien ne soit changé à l'information et à son traitement.

Le terme de RAM (Random Access Memory ou mémoire à accès aléatoire) vient du fait que le temps pour accéder à l'information (le temps qui s'écoule entre le moment où l'on demande une information à la mémoire et celui où elle donne l'information demandée) est indépendant de son adresse: il faut le même temps pour lire une information,

qu'elle soit rangée dans la case d'adresse 7 ou dans celle d'adresse 42153. Toutes les mémoires que nous étudierons, en particulier les mémoires périphériques du Tome IV, n'ont pas cette caractéristique.

En utilisant un micro-ordinateur (par exemple un IBM PC) et le langage Basic, le lecteur pourra directement manipuler ces notions d'adresse, de nom et de valeur de l'information. Soit:

- I% une variable entière, donc le nom de l'information,
- VARPTR(I%) sera alors l'adresse à laquelle sera rangée sa valeur dans la mémoire centrale de l'ordinateur,
- PEEK(VARPTR(I%)) donnera le contenu de la première case affectée à la variable,
- PEEK(VARPTR(I%)+1) donnera le contenu de la deuxième case affectée à la variable I%. Un entier est stocké dans un PC (basé sur un 8088/8086) sur deux octets (l'octet correspond à la case dans cette machine).

Le programme Basic s'écrira ainsi:

```
100 I%=258 : REM valeur de la variable I
110 ADRESSEI=VARPTR(I%)
120 PRINT " Adresse de I% : ";ADRESSEI
130 PRINT " Contenu du premier octet : ";PEEK(ADRESSEI)
140 PRINT " Contenu du second  octet : ";PEEK(ADRESSEI+1)
150 END
```

La valeur 258 affectée à I% en ligne 100 correspond à l'affichage de la valeur 2 par l'instruction PRINT de la ligne 130 puisque dans les 8086 la première case contient les poids faibles du nombre, c'est-à-dire le reste de la division de ce nombre par 256. L'instruction PRINT à la ligne 140 affichera quant à elle 1 puisqu'elle contient les poids forts; ces derniers sont rangés dans le deuxième octet et correspondent à la division entière du nombre par 256 (figure 3e).

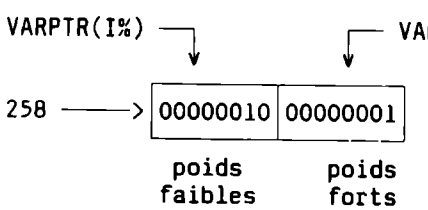


Figure 3e: rangement de la valeur de I% en mémoire.

3.1.5 – Registres.

Un registre est un cas particulier de mémoire. Il en offre toutes les fonctions (lecture, écriture) mais l'information n'y est pas adressée et décomposée à l'intérieur. On dira que l'information de nom X est dans le registre Y, mais on ne dira pas qu'elle se trouve à telle adresse dans le registre Y.

Un registre est une mémoire très petite comparée aux mémoires centrales, mais beaucoup plus rapide (son temps d'accès est plus court).

3.1.6 – Bascules.

Une bascule est un cas particulier de registre: elle ne contient qu'un seul bit. La bascule est donc une mémoire élémentaire, le registre est un ensemble de bascules, et, d'une certaine façon, la mémoire MEV ou MEM est un ensemble de registres, avec en plus la possibilité d'adresser séparément chacune de ses "cases".

3.1.7 – Piles.

Une pile est une mémoire dont seule une case est accessible à un moment donné, de façon implicite, la case dite du haut, aussi appelée **sommet de pile**. Une telle pile est tout à fait analogue à une pile de livres: seul le livre du haut est normalement accessible. Y ranger des livres se dit empiler (push), en enlever se dit dépiler (pull).

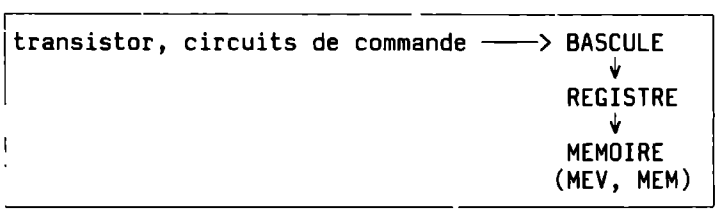


Figure 3f: hiérarchie des mémoires dans le sous-système central.

3.2 – Représentation des informations simples.

Les mémoires vues dans les sections précédentes servent à stocker les informations que le sous-système central doit traiter. Insistons dès maintenant sur le caractère **temporaire** de ce stockage. Les mémoires du sous-système central ne sont là que pour donner les moyens au processeur central d'assurer la fonction de traitement dans des conditions de vitesse correctes. Ce sont principalement des adaptateurs de vitesse entre le processeur central et les mémoires périphériques trop lentes parce que basées sur des techniques faisant intervenir la mécanique.

Il y a d'ailleurs eu des machines sans mémoire centrale, le processeur travaillant directement sur une mémoire périphérique (un tambour sur la CAB500 de la SEA).

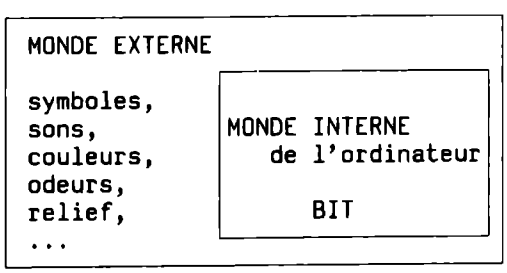


Figure 3g: comparaison entre les informations élémentaires manipulées dans des mondes différents.

3.2.1 – Le bit.

Les informations ne peuvent pas être stockées dans l'ordinateur sous la forme qu'elles prennent dans l'univers des êtres humains. Si ces derniers manipulent des symboles, des couleurs, des sons, des odeurs, du relief, etc, l'ordinateur ne connaît que des informations élémentaires binaires qui n'ont donc que deux valeurs possibles.

3.2.2 – Le caractère.

Toutes les informations stockées dans l'ordinateur sont des agglomérats de bits: le bit étant une information par trop élémentaire pour l'être humain, on oblige en effet l'ordinateur à manipuler simultanément (en parallèle) un certain nombre de bits.

Un des premiers regroupements concerne les caractères: parmi les symboles manipulés par l'être humain figurent les caractères de l'alphabet et les concepteurs d'ordinateur ont très vite décidé que ce dernier devrait manipuler des caractères.

Qu'est-ce qu'un caractère? Pour l'ordinateur un caractère est une séquence de bits, 6 bits autrefois (sextets), 8 bits aujourd'hui (octets). Quelques ordinateurs manipulent simultanément les deux tailles de caractères (DPS8 de Bull), d'autres ont des tailles plus exotiques (7 bits sur PDP10 de DEC, 9 bits sur DPS8). La tendance actuelle est au caractère de huit bits et donc très souvent à des cases de huit bits. Très souvent parce que tous les ordinateurs n'adressent pas leurs cases au niveau du caractère; certains l'adressent par mot, notion que nous verrons ci-dessous.

Les Américains utilisent souvent le terme de byte qui ne correspond pas obligatoirement à la notion d'octet. "Byte" signifie généralement "groupe de bits" et peut tout aussi bien correspondre à un quartet (groupe de 4 bits, appelé aussi "nibble" en américain) qu'à un sextet, à un octet ou à un nonet (9 bits). Lorsqu'ils veulent être précis, les auteurs anglo-saxons emploient le terme français d'octet qui signifie exactement huit bits.

3.2.2.1 – Représentations hexadécimale et octale.

Lorsque l'on désire représenter le contenu d'une case de 8 bits, il est fastidieux d'aligner les 8 bits. On les découpe en deux fois quatre bits, quatre à gauche (quartet de gauche) et quatre à droite, et l'on représente chacun de ces deux quartets par le chiffre correspondant en base 16. On réalise ainsi un changement de base, de la base deux à la base seize.

Les chiffres hexadécimaux sont donc constitués par les chiffres décimaux de 0 à 9, puis par les lettres A à F pour les chiffres de valeur 10 à 15.

Le **décimal codé binaire** (DCB, BCD en américain pour binary coded decimal) est un code dans lequel chacun des chiffres décimaux est codé sur 4 bits comme le montre le tableau 3h.

La représentation **octale** consiste à regrouper les bits par trois afin de représenter les huit chiffres de zéro à sept. Un octet sera

ainsi représenté par trois chiffres octaux. En langage C, '\153' correspondra au message binaire 0110 1011, c'est-à-dire au nombre décimal 107.

0000	0	0100	4	1000	8	1100	C	*Tableau 3h: correspondance entre quartet et chiffre hexadécimal.*
0001	1	0101	5	1001	9	1101	D	
0010	2	0110	6	1010	A	1110	E	
0011	3	0111	7	1011	B	1111	F	

3.2.2.2 – Codage des caractères.

Une fois choisi le nombre de bits pour stocker les caractères, il reste encore une décision à prendre: comment **coder** chacun de ces caractères, c'est-à-dire quelle configuration de bits associer à chaque caractère.

Un octet, soit huit bits, offre 256 (deux à la puissance huit) configurations de bits différentes. Deux grands choix (codes) dominent actuellement:

- le code EBCDIC (Extended Binary Coded Decimal Interchange Code) apparu avec la série 360 d'IBM. Il code les caractères sur huit bits,
- le code ASCII (American Standard Code for Information Interchange) plus ancien, que l'on trouve dans la grande majorité des ordinateurs non compatibles avec la série 360 d'IBM. Il code les caractères sur 7 bits (soit 128 combinaisons différentes). La plupart des ordinateurs utilisant des octets, un des bits est inutilisé; il est généralement systématiquement mis à zéro, quelques ordinateurs (cas de PRIME) le mettent à un. Certaines machines ont "étendu" de façon arbitraire le code ASCII sur 7 bits et utilisent les 8 bits; c'est le cas du PC par exemple.

Les codages télétex (T.61) et vidéotex (T.100) utilisent à la base un codage ASCII.

AT&T développe actuellement une nouvelle norme permettant de représenter les idéogrammes chinois sur 16 bits, en conservant la compatibilité ASCII dans la mesure où chaque octet a son bit de gauche égal à zéro.

Si nous reprenons le Basic, l'instruction **PRINT ASC("A")** affichera 65, ce qui correspond à 41 en hexadécimal, soit 01000001 en binaire, c'est-à-dire le code ASCII de la lettre A en majuscule.

Nous nous bornerons à ce rapide aperçu en ce qui concerne les codes car il s'agit d'un vaste sujet qui justifie un ouvrage à lui seul.

3.2.2.3 – Chaînes de caractères.

On appelle chaîne de caractères une séquence de caractères consécutifs en mémoire. A l'époque des caractères codés sur 6 bits il existait des machines (H2000 d'Honeywell) dans lesquelles un des bits de l'octet matérialisait la fin d'une chaîne de caractères. A l'heure actuelle il n'y a généralement rien qui indique en mémoire la fin des chaînes de caractères: ce sont les instructions qui précisent la longueur des chaînes.

Cependant UNIX et le langage C délimitent les chaînes de caractères par un caractère final ayant ses huit bits à zéro ('\0' en C) et il semblerait qu'AT&T traite ainsi les chaînes de caractères au niveau des instructions machine de son microprocesseur. Certains interpréteurs de Basic gèrent aussi de cette façon les chaînes de caractères par un caractère spécial à la fin. Ce caractère de bornage ne peut plus être représenté à l'intérieur de la chaîne.

3.2.3 – Les nombres.

En dehors des caractères, l'ordinateur manipule des nombres. Ces nombres sont stockés sous diverses représentations.

3.2.3.1 – Représentations décimales.

Les représentations décimales des nombres permettent de rester en base 10 à l'intérieur de l'ordinateur: chaque chiffre décimal est représenté séparément par une séquence (une chaîne) de bits. On distingue les représentations en:

- **décimal étendu** : chaque chiffre du nombre est considéré comme un caractère et nous retrouvons les chaînes de caractères vues ci-dessus. On parlera aussi de "chaînes numériques";
- **décimal condensé** : comme il n'y a que 10 chiffres en base 10, il est luxueux d'utiliser les 8 bits du codage caractère pour chacun des chiffres; quatre bits suffisent: chaque octet contient alors deux chiffres, il en résulte un gain de place en mémoire par rapport au décimal étendu. Il est souvent appelé BCD (Binaire Codé Décimal, DCB en américain) dans le monde des microprocesseurs. Ce mode de codage n'existe pas dans les machines à caractères codés sur 6 bits;

La plupart des machines n'utilisent la représentation décimale que pour des entiers. Cependant la représentation des nombres décimaux (non entiers) est possible dans quelques ordinateurs et l'on parle alors de **décimal flottant**. C'est le cas des DPS8 de Bull.

3.2.3.2 – Représentations binaires.

L'ordinateur est naturellement orienté vers la base 2 puisque ses informations élémentaires, les bits, ne prennent que deux valeurs: on conçoit donc que la représentation des nombres qui lui est la plus naturelle est la représentation des nombres en base 2. Le nombre 13 en base 10 sera stocké dans l'ordinateur, à l'aide de 4 bits, sous la forme 1101 en base 2. En réalité, contrairement aux chaînes de caractères dont la longueur est variable, les nombres en binaire sont représentés sous un **format fixe**. Le nombre de bits associés à un nombre en binaire est constant quelque soit la valeur du nombre. Cette quantité fixe de bits pour tout nombre est appelée un **mot**.

La taille du mot (son nombre de bits) varie selon les ordinateurs: 8 bits sur les microprocesseurs dits 8 bits comme le 6502, le 8080, le Z80, etc. Il pourra atteindre 64 bits sur les gros calculateurs comme le CRAY1. Si sur les premiers ordinateurs orientés vers le calcul le mot était la seule structure physique dans laquelle on rangeait les nombres, il y a à l'heure actuelle des ordinateurs offrant

la possibilité de manipuler des:
- quarts de mot ou même moins,
- demi-mots,
- quadruples mots, doubles mots.

Pour représenter les nombres qui ne sont pas des entiers on adopte la représentation en **virgule flottante** (floating point en américain puisque le point remplace la virgule aux Etats-Unis). Chaque nombre y est représenté sous forme d'un couple [exposant, mantisse]. La mantisse, toujours inférieure à 1, donne les chiffres significatifs (d'où le terme de précision que l'on trouve dans les instructions DOUBLE PRECISION du Fortran). L'exposant donne la magnitude.

Par exemple, le nombre 43527 en base 10, s'écrira

$$10^5 * 0{,}43527$$

en virgule flottante. 5 représente l'exposant et 43527 la mantisse. Si on dispose de 8 positions pour stocker les nombres flottants on obtiendra la représentation donnée figure 3i.

0 5	4 3 5 2 7 0

Figure 3i: représentation en flottant base 10 de 43257 avec 2 positions pour l'exposant et 6 pour la mantisse.

En restant en base 10, le nombre 43,527 donnera la représentation de la figure 3j.

0 2	4 3 5 2 7 0

Figure 3j: représentation en flottant base 10 de 43,257 avec 2 positions pour l'exposant et 6 pour la mantisse.

Le nombre de bits affectés à l'exposant ainsi que le nombre de bits affectés à la mantisse sont fixes pour une machine donnée. On trouve généralement 8 bits pour l'exposant et 24 bits pour la mantisse sur une machine à mots de 32 bits.

Lorsque l'on utilise des doubles ou quadruples mots, la taille de la mantisse augmente en conséquence. Dans certains ordinateurs la taille de l'exposant augmente aussi (1100 de Sperry).

Si la mantisse est toujours représentée en base 2 avec signe séparé, l'exposant l'est en base 2 ou en base 16 selon les ordinateurs. Le DPS8 de Bull offre maintenant les deux possibilités depuis qu'Honeywell a repris le parc des SIGMA de XDS (ex SDS). Ainsi

0 1	1 0 0 0 0 0 0
E	M

correspondra à $2^1 * 2^{-1}$

soit 1 si l'exposant est interprété en base 2, alors qu'il correspondra à

$16^1 * 2^{-1}$

soit 8 si l'exposant est interprété en base 16.

Une base faible favorise la précision (plus grand nombre de chiffres significatifs), une base élevée offre de plus grandes magnitudes.

Le dernier point que nous traiterons au sujet des représentations en flottant est la **normalisation**. Une représentation est normalisée si le premier chiffre de la mantisse n'est pas nul:

- le premier bit vaut 1 si l'exposant est interprété en base 2,
- les quatre premiers bits ne sont pas tous égaux à zéro si l'exposant est interprété en base 16.

La représentation normalisée permet d'obtenir le nombre maximal de chiffres significatifs. Certains ordinateurs travaillent systématiquement en normalisé, d'autres non (cas des DPS8). Dans le second cas il faut des instructions de normalisation. Dans le premier cas il est possible de ne pas stocker le premier bit de la mantisse puisque l'on sait (si l'exposant est en base 2) qu'il est nécessairement égal à un, d'où un gain de précision (cas des PDP11 et VAX de DEC où il est appelé "bit caché", hidden bit en américain).

Le lecteur désireux de mettre en pratique ces notions pourra reprendre l'exemple donné au paragraphe 3.1.4 avec le programme Basic ci-dessous pour un micro-ordinateur du type PC d'IBM:

```
10 DEFSNG S : REM VARIABLE SIMPLE PRECISION
20 INPUT "S = ",S
30 FOR I=0 TO 3
40   PRINT HEX$(PEEK(VARPTR(S)+I))
50 NEXT I
60 GOTO 20
```

Ce programme affichera en hexadécimal le contenu des 4 octets affectés par ce BASIC à une variable flottante en simple précision. Le lecteur essaiera plusieurs valeurs comme 0 -0 1 2 3 4 5 6 7 8 16 32 0,25 0,125 etc, -1 -2 -3 -4 -8 etc.

S'il fait tourner ce programme sur plusieurs systèmes (en passant d'un IBM PC sous MS-DOS à un MICRAL de Bull sous CP/M-86 par exemple) il pourra observer des variations (0 et -0 ont des représentations identiques sur PC, différentes sur MICRAL).

Les coprocesseurs numériques des microprocesseurs suivent généralement la norme IEEE-P754.

3.3 – Résumé.

Nous avons abordé dans ce chapitre, les notions de:

- **mémoire**: circuit conservant l'information durant un certain temps,
- **octet**: groupe de 8 bits contenant un caractère;
- **byte**: groupe de n bits, couramment 7, 8 ou 9, contenant un caractère;

- **mot**: groupe de n bits contenant un nombre, couramment 32;
- **caractère** avec son acception habituelle.

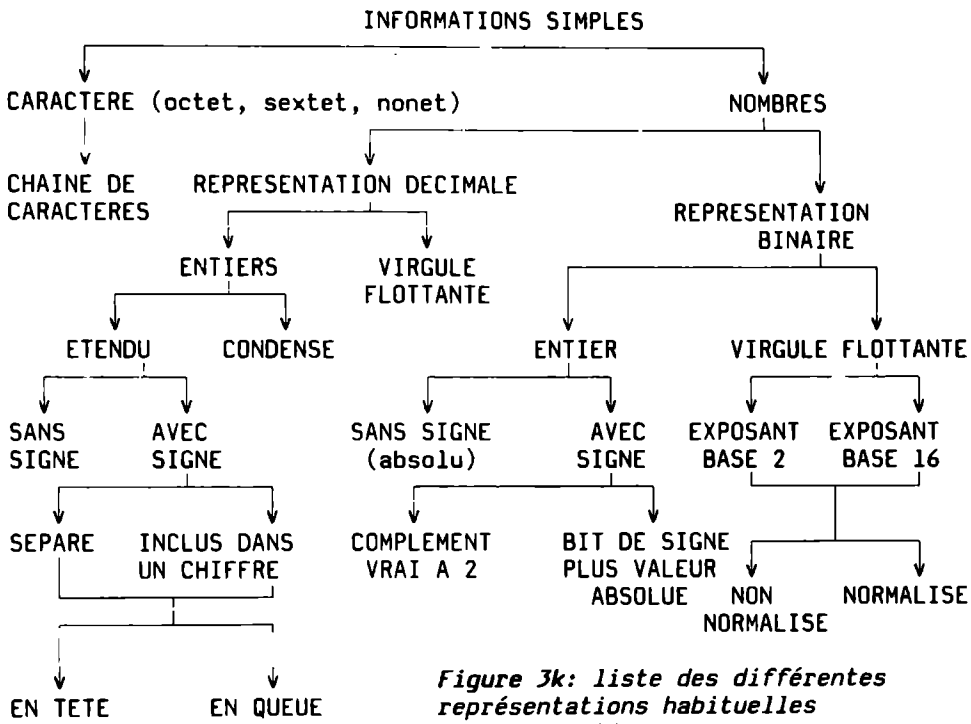


Figure 3k: liste des différentes représentations habituelles des informations.

chapitre 4

les instructions

4.1 – Rôle des instructions "machine".

Un sous-système central, au niveau du matériel, ne sait faire qu'une seule chose: exécuter des instructions à la suite de l'arrivée d'un événement initial (une interruption en général).

Ces instructions ne sont pas celles auxquelles le lecteur est éventuellement accoutumé: ce ne sont pas les instructions des langages évolués comme Basic, Cobol, Fortran, PL/I, Pascal, APL, C, Lisp, etc. Ce sont les instructions que sait directement exécuter (on dira interpréter aussi) le matériel, sans l'aide d'aucun logiciel, que ce soit un compilateur ou un interpréteur. On les appellera des **instructions machine**.

L'ensemble des instructions comprises par un ordinateur donné constitue le **langage machine** de cet ordinateur, aussi appelé **jeu d'instructions**.

Les langages d'assemblage sont ceux qui se rapprochent le plus du langage machine puisque l'on peut les représenter comme un sur-ensemble du langage machine (cf Tome V).

Ces instructions matérialisent dans l'ordinateur sa fonction de **traitement** de l'information: toute application, tout problème soluble, tout algorithme se réduira à un moment donné (voir figure 4a) à une **séquence** d'instructions qu'interprétera le matériel. Pour lui tout est séquence d'instructions.

4.2 – Structure des instructions.

4.2.1 – Structure théorique.

Une instruction doit préciser:

- l'opération à effectuer: c'est le rôle de la partie **code opération** de l'instruction qui indique ("code") l'opération; on trouvera l'abréviation "code-op" dans de nombreux schémas,
- les **données** sur lesquelles s'effectue l'opération en question,
- l'adresse de **l'instruction suivante**.

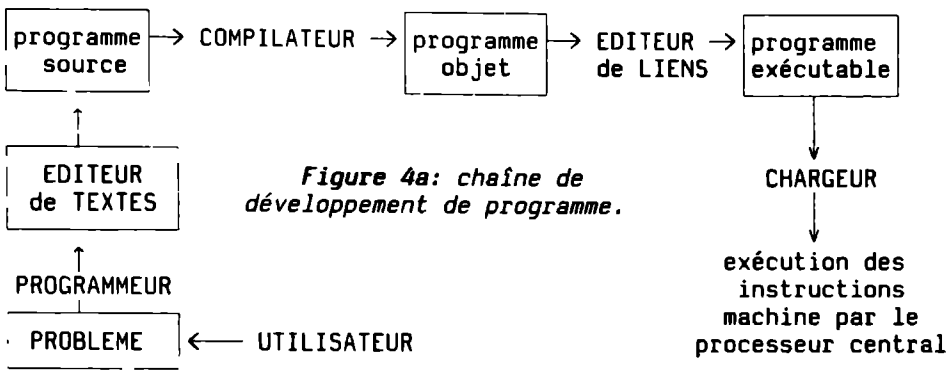


Figure 4a: chaîne de développement de programme.

En fait aucun ordinateur ne possède aujourd'hui de telles instructions. L'ordinateur fait une hypothèse de séquentialité du rangement des instructions: la troisième partie devient inutile et l'instruction ne contient pas l'adresse de l'instruction suivante. L'adresse de l'instruction suivante est donc égale à l'adresse de l'instruction courante plus 1 (ou plus n si l'instruction courante occupe plusieurs cases de mémoire comme dans nombre d'ordinateurs actuels). Des trois parties qui se trouvent dans la figure 4b, seules les deux premières sont conservées.

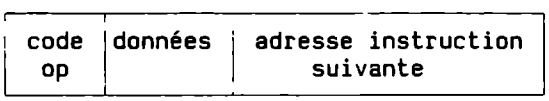


Figure 4b: structure théorique de l'instruction.

Un nouveau type d'instruction est alors nécessaire: les instructions de **branchement** qui permettent de ne pas exécuter systématiquement toutes les instructions rangées en séquence dans la mémoire (figure 4c). La condition que l'on voit dans la figure 4c permet de disposer d'instructions de branchement conditionnel que nous expliciterons plus loin.

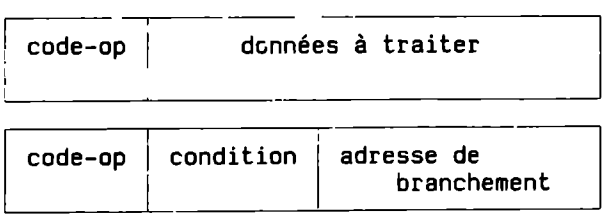


Figure 4c: deux types d'instruction introduits par l'hypothèse de non séquentialité.

4.2.2 – Structure réelle.

La partie "données" de la figure 4c permet à l'instruction de connaître les données à traiter. Les informations nécessaires à ce traitement sont:

- le type de la donnée: bit, caractère, entier, flottant, décimal étendu, signé ou non, etc.,
- la structure de la donnée: scalaire, vecteur, matrice, etc,
- la longueur de la donnée: chaîne, mot, double mot, etc.

Ces informations appartiennent à la donnée, elles n'appartiennent en rien a priori à l'instruction. Il est donc logique de lier ces caractéristiques à la valeur de la donnée en mémoire, comme le montre la figure 4d.

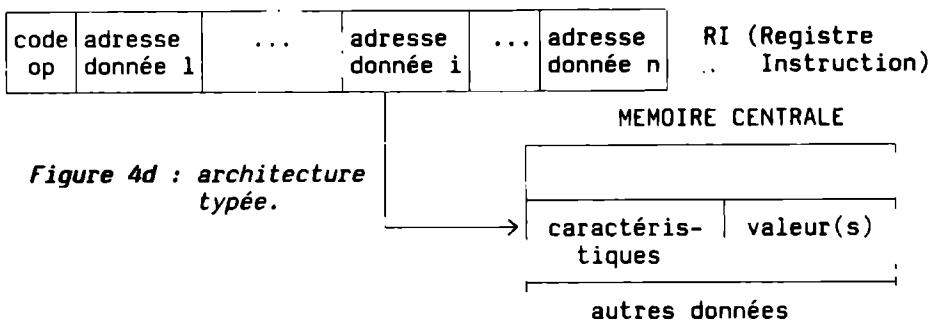


Figure 4d : architecture typée.

En fait très peu d'ordinateurs commercialisés ont une structure conforme à celle de la figure 4d. Il n'y a guère que les séries B6700, B6800, B6900 de Burroughs, pour lesquelles on parle d'architecture typée (tagged architecture, la zone de mémoire qui contient les caractéristiques d'une donnée est appelée "tag"), qui possèdent cette architecture. La figure 4e donne une idée de quelques caractéristiques que l'on trouve sur ces ordinateurs Burroughs.

```
544 4444444433333333332222222222211111111111          bits du
098 76543210987654321098765432109876543210987654321O    mot
```

tag	valeur de la donnée

TAG (accessible en mode privilégié seulement)
000 opérande simple précision
001 indirection
010 opérande double précision
100 indexation
101 descripteur

Figure 4e: tags dans les grands et moyens systèmes de Burroughs.

Dans la plupart des ordinateurs, les caractéristiques des données (on dira aussi opérandes) sont stockées dans l'instruction et non pas avec la valeur. Elles figureront donc autant de fois en machine qu'il y a d'instructions manipulant la donnée. Ces caractéristiques seront très souvent mélangées avec le code opération, totalement ou partiellement. C'est ainsi que sur un 370 d'IBM il n'y a pas moins de 15

codes opération différents pour l'addition:
- addition entière de demi-mots (16 bits) signés,
- addition entière de demi-mots non signés,
- addition entière de mots,
- addition en flottant simple précision,
- addition en flottant double précision,
- addition en flottant quadruple précision (REAL*8 en Fortran soit 128 bits),
- addition en décimal condensé (avec au plus quinze chiffres),
- etc.

Et encore, la liste pourrait être plus longue: il n'y a pas d'addition en décimal étendu.

La valeur d'une donnée étant séparée de ses caractéristiques, on peut très bien sur ce genre de machine interpréter différemment une même valeur: une fois en tant que valeur d'un nombre entier avec une instruction d'addition entière, une autre fois en tant que chaîne de caractères, une autre encore en tant que premier mot d'un nombre en flottant double précision. C'est ce que fait le programme en C ci-dessous:

```
main()
    {
     int i,entier[10];
     for (i=0;i<11;i++)                              /* init.    */
         entier[i]=i;                                /* tableau  */
     spch(entier);                                 /* appel des    */
     spdp(entier);                                 /* sous-progr.  */
     exit(0);
    }
spdp(param)                     /* incrémente le début du tableau  */
    double *param;              /* considéré comme nombre flottant */
    { (*param)++; return(0); }
spch(param)                     /* le tableau est vu comme une     */
    char param[];               /* chaîne de caractères            */
    int i;
    { for (i=0;i<20;i++) param[i]=param[20-i]; return(0); }
```

Le code opération peut aussi impliquer un mode d'adressage (chapitre 7): il différera par exemple selon que les opérandes sont dans des registres ou dans la mémoire centrale. Ainsi la série 360/370 d'IBM dispose de l'instruction "A" pour additionner deux opérandes, l'un étant dans un registre, l'autre en mémoire centrale. Si les deux opérandes sont dans des registres il faudra utiliser l'instruction "AR".

Ces noms "A" et "AR" sont des noms symboliques que l'on donne aux codes opération des instructions, permettant d'en parler de façon plus pratique. Il est effectivement plus difficile de parler de l'instruction 01001100 que de JMP sur un 6502 (cf annexe A). On retrouve ces noms (ces "mnémoniques") dans les langages d'assemblage.

Nous venons donc de voir que les instructions contiennent pour chacun de leurs opérandes:
- l'adresse,
- généralement les caractéristiques.

Nous avons supposé que toutes les données étaient accessibles pour n'importe quelle instruction. Ceci est vrai sur les machines simples comme les 6502, mais l'est de moins en moins quand le niveau de l'architecture augmente:
- toutes les instructions ne peuvent alors plus produire les adresses de toutes les cases de la mémoire centrale,
- certaines données sont protégées et y accéder nécessite la possession de droits adéquats (figure 4f).

Nous aborderons ces notions plus loin.

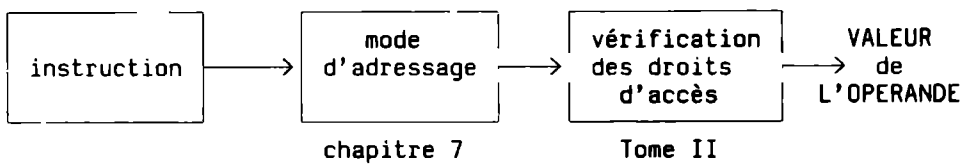


Figure 4f: chemin entre l'instruction et l'opérande.

4.3 – Classification des instructions.

Le jeu d'instructions d'un ordinateur contient des instructions très diverses. Nous avons déjà vu la séparation entre instructions de traitement des données et instructions de branchement. Les premières sont porteuses de la fonction de traitement que veut réaliser l'utilisateur. Les secondes ne sont là que parce que la structure de l'ordinateur l'impose: "inutiles" pour l'utilisateur, elles ne produisent aucun résultat significatif; on parlera d'instructions **non fonctionnelles**. Nous préciserons ultérieurement cette classification synthétisée figure 4h.

4.3.1 – Instructions fonctionnelles.

Cette catégorie comprend les instructions suivantes:

- **arithmétiques** : addition (ADC sur 6502), soustraction (SBC), multiplication, division, changement de signe, incrémentation (ajouter 1, INC sur 6502), décrémentation (DEC sur 6502), décalage arithmétique (en tenant compte du signe);
- **logiques** : rotations et opérations booléennes:
  * **décalages** à droite (LSR sur 6502), à gauche (ASL sur 6502), de un ou plusieurs bits selon les ordinateurs (un seul sur le 6502),
  * **rotations** à droite (ROR sur 6502), à gauche (ROL sur 6502),
  * **opérations booléennes** comme le ET (AND sur 6502), le OU (ORA sur 6502), le OU EXCLUSIF (EOR sur 6502), le NON (sur le 8086 par exemple).

- de comparaison :
  * de mots (CMP sur 6502),
  * de chaînes de caractères (CLC sur IBM 370),
  * de bit; quelques ordinateurs disposent d'instructions pouvant tester n'importe quel bit d'un mot; dans le cas général le test d'un bit particulier se déroule en deux étapes: dans un premier temps on isole le bit à tester (masquage par un ET, décalage, etc.), et ensuite on regarde si le résultat est nul ou non;
  * de chaînes de bits.
- traitement de chaînes: outre les copies (de gauche à droite, et de droite à gauche dans certains ordinateurs), et les comparaisons, on trouve des instructions de recherche de sous-chaînes (parfois limitées à un caractère comme dans le 8086) qui correspondent à l'instruction Basic INSTR. La recherche en arrière est parfois disponible (cas des NS16000);
- traitement de listes (list processing) appelées "queues" en américain. On trouve des instructions de rangement dans la liste, en tête (ENQH sur MV), en queue (ENQT sur MV), d'extraction de la liste (DEQUE sur MV), de recherche, en avant et en arrière.

4.3.2 – Instructions non fonctionnelles.

Nous classerons comme suit les instructions non fonctionnelles:

- celles de transfert qui sont dues aux différents types de mémoire de l'ordinateur.Nous trouverons ainsi:
  * les instructions de chargement (load en américain) qui transfèrent le contenu d'un mot de la mémoire centrale dans un registre du processeur (LDA sur 6502), ou qui transférent de la pile vers un registre (PLA sur 6502),
  * les instructions inverses qui rangent (store en américain) le contenu d'un registre dans la mémoire centrale (STA sur 6502), ou dans la pile (PHA sur 6502),
  * les instructions de transfert entre registres (transfer en américain, TXA sur 6502), ou entre "cases" de la mémoire centrale (move en américain),
  * les instructions d'entrée/sortie (input/output, I/O en américain) comme les IN et OUT des 8086/8088.

  L'étude de [MD084] sur la fréquence des instructions machine 370 exécutées dans le cas de programmes COBOL, montre que les chargements de registre sont les instructions les plus utilisées: l'instruction L (load) vient en première position avec une fréquence de 17,1 %, les LA (chargement d'adresse) en cinquième position avec 4,5 %, les LH (chargement d'un demi-mot) en huitième position avec 3,7 %, soit plus de 25 % pour ces seules instructions. Si l'on ajoute les rangements (ST), les transferts entre registre (LR), on arrive à près d'un tiers des instructions exécutées;
- celles de conversion entre les différentes représentations des données:
  * d'entier court à entier long comme CBW sur 8086 pour passer de 8 à 16 bits ou CWD pour passer de 16 à 32 bits,
  * d'entier à flottant,
  * d'entier à décimal, comme CVD sur 360/370 pour passer en décimal

condensé (la conversion d'entier en décimal étendu n'y est pas possible directement et inutile puisque les calculs ne peuvent se faire qu'en condensé sur cet ordinateur),
* de décimal étendu à décimal condensé (PACK sur 360/370),
* de caractères 6 bits en caractères 8 bits (MLR sur DPS8),
* de flottant non normalisé en flottant normalisé (FNR sur DPS8),
* de traduction comme XLAT sur 8086,
* d'édition (ED),
* etc.

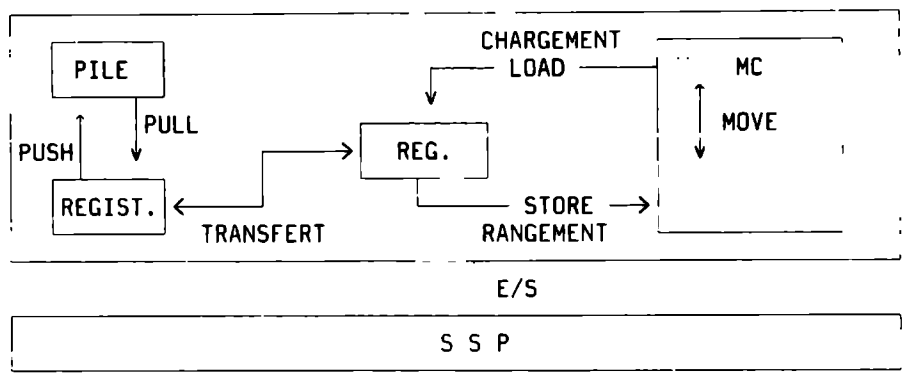


Figure 4g: les instructions de mouvement entre mémoires.

- celles dites **procédurales** parmi lesquelles on trouve:
  * les **branchements** appelés "branch", "jump", "skip" en américain:
    . inconditionnels (JMP sur 6502),
    . conditionnels (BNE sur 6502), les conditions étant généralement définies par l'exécution d'une instruction arithmétique et matérialisées par des **indicateurs** (code condition sur 360/370 d'IBM) que nous verrons dans un prochain chapitre;
    . multiples qui correspondent aux CASE OF de Pascal, GOTO indexé du Fortran, au ON I GOTO du Basic, au SWITCH du C, etc.

    On trouve aussi des instructions qui correspondent exactement au SI (IF) du Fortran, avec les trois possibilités, inférieur, égal, supérieur (série 16 de Bull). Une étude de [MD084] sur les instructions 370 exécutées par le système d'exploitation MVS SP1.3 montre que BC (branchement conditionnel) est la plus utilisée, les branchements comptant pour un tiers des instructions exécutées, 89 % d'entre eux étant des branchements conditionnels;
  * les **boucles** (loop en américain): LOOP sur 8086; RPT (répétition) sur DPS8 permet avec sa variante RPL de répéter l'exécution d'une liste chaînée d'instructions,
  * les **appels** de sous-programmes (JSR sur 6502) et retour vers l'appelant (RTS sur 6502). Il convient de distinguer plusieurs niveaux d'instruction d'appel:

. le premier niveau correspond à l'appel d'une séquence quelconque d'instructions et peut se trouver dans des machines très simples comme le 6502;
. le deuxième niveau se trouve dans les machines d'architecture plus complexe comme les VAX (CALLS et CALLG), les DPS7 (PRSK, ENTER), les DPS8 sous GCOS8, les PRIME, les iAPX 286 (CALL), les iAPX 432 (CALL CONTEXT), etc. L'appel de deuxième niveau correspond à des appels de sous-programmes généralement compilés séparément ou n'étant pas accessibles à tous les utilisateurs;

- celles de **synchronisation** comme:
  * l'instruction "test and set" qui s'exécute de façon non interruptible, aussi bien au niveau du processeur que des accès mémoire consécutifs qu'elle peut provoquer. Elle permet ainsi de tester le contenu d'une zone mémoire et de le modifier s'il n'est pas nul. C'est le cas de TAS sur le 68000 de Motorola;
  * les instructions portant sur les sémaphores. Cas des DPS7, PRIME;
- celles de **gestion** de l'ordinateur comme
  * **l'arrêt** de l'ordinateur (HLT sur 8086); tous les ordinateurs n'ont pas cette instruction,
  * le changement de **mode de fonctionnement**. Les ordinateurs simples comme le 6502 et même le 8086 n'ont qu'un seul mode de fonctionnement. Si l'on considère un 360 d'IBM ou un DPS8 de Bull sous GCOS3 on a déjà deux modes de fonctionnement: dans l'un l'ordinateur accepte d'exécuter toutes les instructions, et l'on parlera alors, selon les constructeurs, de mode maître (ou système, superviseur, privilégié, etc.); dans l'autre mode l'ordinateur n'exécutera qu'un sous-ensemble du jeu d'instructions et l'on parlera alors de mode esclave (ou problème, utilisateur, normal, etc.); c'est dans ce dernier mode que se dérouleront généralement les instructions des programmes d'un utilisateur, le premier étant réservé au système d'exploitation. Si l'on considère maintenant un DPS8 sous GCOS8 on a trois modes de fonctionnement. Dans les DPS7, PRIME, VAX, MV, iAPX 286, on passe à quatre modes de fonctionnement appelés des **anneaux** puisque les pouvoirs des modes sont croissants. D'autres systèmes (MULTICS, série 2900 d'ICL, CYBER 180) offrent un plus grand nombre d'anneaux. Le passage entre modes fait l'objet d'une instruction spéciale (SVC sur 360, MME sur DPS8, ER sur 1100 de Sperry) ou bien est intégré au mécanisme d'appel de sous-programme (CALL sur DPS7, cas aussi des MV de Data General). Nous reviendrons plus loin sur cet aspect qui est lié aux problèmes de sécurité et de sûreté de fonctionnement de l'ordinateur,
  * le positionnement des **indicateurs** (SED, CLI sur 6502),
  * et plus généralement les instructions gérant les objets qui ne sont pas vus de l'utilisateur "normal", comme l'activation des antémémoires (chapitre 18), celle des registres de traduction (Tome II), la gestion des registres de contrôle (CRi sur 370 pour gérer la pagination), etc,
- celles que l'on ne sait pas où classer comme:
  * l'instruction EXECUTE qui permet d'exécuter une autre instruction en en modifiant quelques bits (par exemple en changeant la longueur d'une copie de chaînes, dans le cas des 360),

* les instructions liées aux adresses comme LEA sur 8086,
* les instructions testant le dépassement d'indice dans un tableau (CHK sur 68000, INDEX sur VAX),
* les instructions réservant et libérant l'espace alloué aux variables dynamiques (cf. Tome V) comme LINK et UNLK sur 68000,
* instruction d'aide à la mise au point (debug) comme BRK sur 6502,
* etc.

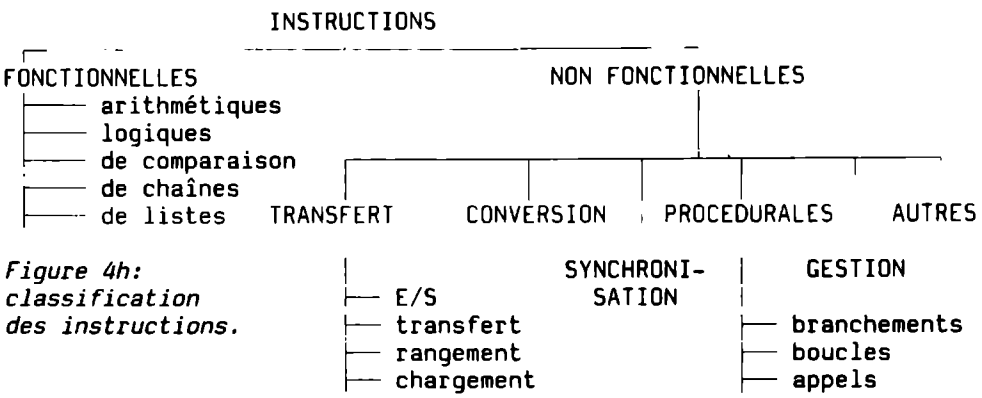


Figure 4h: classification des instructions.

4.3.3 – Remarques sur ce classement.

Le classement en instructions fonctionnelles et non fonctionnelles peut être discuté, d'autant plus qu'une même instruction pourra être utilisée, de façon fonctionnelle ou non, selon le cas. C'est cependant un point de vue intéressant qui permet d'insister sur la grande différence, entre ce que veut l'utilisateur humain et ce que lui impose la machine, du fait de son "bas" niveau structurel.

PROCESSEUR	"IDEAL"	IBM 7090	IBM 360	DEC PDP10	DEC PDP11
M/F	0,0	2,0	2,9	1,5	2,6
P/F	0,0	0,8	2,5	1,1	3,7
(M+P)/F	0,0	2,8	5,5	2,6	6,3

***Tableau 4i:** exemples de proportions d'instructions non fonctionnelles.*

M = instructions de mouvement dans les mémoires (load, store)
P = instructions procédurales (test, branchement, comparaison)
F = instructions fonctionnelles (arithmétiques, logiques)

On trouve dans [FLY80] une étude de l'efficacité des jeux d'instructions sous l'aspect du rapport entre nombre d'instructions fonctionnelles et non fonctionnelles dans des programmes de référence. Le tableau 4i en donne quelques résultats.

4.4 – Caractéristiques des jeux d'instructions.

Outre les considérations d'efficacité en nombre d'instructions et en taille mémoire occupée abordées ci-dessus, d'autres critères permettent de comparer les différents jeux d'instruction.

4.4.1 – Taille du jeu d'instructions.

Il est intéressant de rappeler que la théorie des automates fixe un nombre minimal d'instructions pour obtenir un ordinateur universel. Ce nombre est égal à deux et correspond à:

- une première instruction qui décrémente et réalise un branchement si zéro est atteint,
- une seconde instruction d'incrémentation.

Les petits ordinateurs ont généralement quelques dizaines d'instructions et au moins une centaine de codes opération:

PROCESSEUR	6502	4004	8008	8080	Z80	8086	Z8000	68000	NS16032
NBR. INST.	<50	46	48	78	158	111	110	56	86
NBR. CODOP.	151				696		410		>1000

Les minis, moyens et gros ordinateurs ont généralement quelques centaines d'instructions: 208 sur 370, 214 sur HP3000/42.

Il n'est pas toujours aisé de différencier nombre d'instructions et nombre de codes opération. Pour illustration prenons l'instruction ADC (addition) du 6502: elle a 8 codes opération différents selon que l'opérande se trouve dans l'instruction elle-même, dans les 256 premiers octets de la mémoire, etc. Toujours dans le cas du 6502 faut-il compter CLC, CLD, CLI, CLV comme une ou quatre instructions? Toutes remettent à zéro l'un des indicateurs.

Les ordinateurs ont eu tendance à avoir de plus en plus d'instructions. On peut le constater avec les microprocesseurs d'INTEL dont le nombre d'instructions est passé de 48 pour le 8008 à 111 sur le 8086.

On a vu apparaître ces dernières années de nouvelles machines dont une des caractéristiques est d'avoir un nombre d'instructions volontairement petit. Ce sont les ordinateurs dits **RISC** (Reduced Instruction Set Computer: ordinateur à jeu d'instructions réduit), résultats d'études menées à l'université de Berkeley en Californie. Ces études conduisent à penser qu'un processeur avec un nombre minimum de modes d'adressage et d'instructions référençant la mémoire centrale serait tout à fait adapté à la compilation et à l'exécution de code produit par les langages évolués. Cette approche conduit à un grand nombre de registres dans le processeur.

Bull commercialise une machine RISC, le SPS9 conçu par le constructeur américain RIDGE. Les Pyramid 90X sont aussi basés sur un processeur RISC, ainsi que le HCX-7 de Harris, les 6150 et 801 (non commercialisé) d'IBM.

4.4.2 – Longueur des instructions.

Historiquement les ordinateurs ont commencé avec des instructions de longueur fixe, un mot en général. Ceci était dû au fait que la logique était rudimentaire et que le processeur ne savait charger qu'un mot. La plupart ont aujourd'hui des instructions de longueur variable (un, deux ou trois caractères sur le 6502).

4.4.3 – Orientation mémoire, pile, registres.

Certains concepteurs ont des opinions très précises en ce qui concerne l'utilisation de la mémoire centrale, de registres et de piles.

Certaines machines ont une orientation **registre** très nette. C'est le cas des CYBER de CDC (Control Data Corp.), les opérations ne se font que sur registres, aussi bien les opérandes source que les résultats doivent se trouver dans des registres. Il y faut bien entendu des chargements/rangements entre mémoire et registres, déclenchés dans le cas de CDC par l'utilisation des registres adresse, mais aucun calcul ne se fait avec un opérande en mémoire. L'orientation registre est une caractéristique des ordinateurs orientés vers le calcul scientifique.

Dans les ordinateurs orientés **mémoire**, il n'y a pas de registre visible de l'utilisateur, tout semble se faire en mémoire. C'est le cas de l'iAPX 432 d'INTEL, du HP9000, de l'IBM 38.

Dans les machines orientées **pile**, toutes les opérations se font sur des opérandes rangés dans une pile. Ainsi l'addition se réalisera en additionnant les deux nombres en haut de la pile, ces deux nombres disparaîtront du haut de la pile et le résultat deviendra le nouveau sommet de la pile, comme le montre la figure 4j.

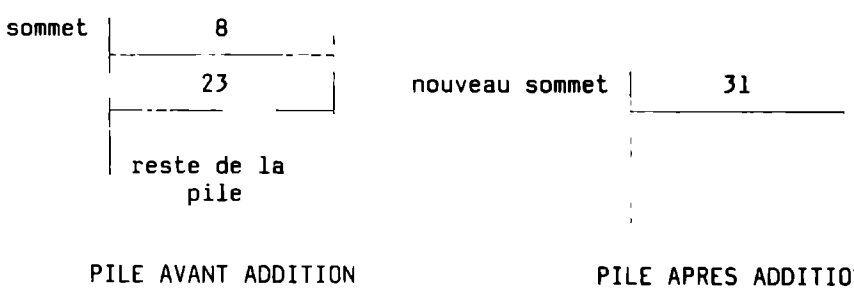


Figure 4j: addition de deux nombres avec une pile.

Ce fonctionnement est identique à celui des calculettes vendues par HP et utilisant la notation polonaise. Les ordinateurs orientés pile (pile de calcul pour être précis) sont ceux commercialisés par Burroughs, par HP sur les HP3000, par Tandem; ce n'est pas étonnant puisque ce sont des transfuges de ces sociétés qui, chaque fois, ont conçu ces machines. Le coprocesseur 8087 d'INTEL permet aussi de traiter les 8 registres comme une pile de calcul.

Dans la pratique de nombreux processeurs sont hybrides, ils n'appartiennent à aucune de ces catégories. Ainsi les 370 opèrent sur des entiers qui se trouvent dans les registres, mais l'un des deux opérandes peut aussi se trouver en mémoire centrale. Le résultat par contre ira toujours dans un registre.

Dans les DPS7, DPS8, ou même dans le 6502 pour certaines opérations, les résultats peuvent être directement rangés en mémoire, sans passer par des registres. Dans les 370, les DPS7 ou DPS8, les instructions manipulant des données décimales travaillent toujours en mémoire centrale.

4.4.4 – Jeu d'instructions symétrique.

Un jeu d'instructions est symétrique quand chaque opération dispose de sa symétrique: il y a addition et soustraction, il y a multiplication et division, etc. Le 6502, le Z8000 ont des jeux d'instructions symétriques, celui du 8080 ne l'est pas pour l'addition.

Cette caractéristique des jeux d'instructions n'est pas essentielle. Nous l'avons citée afin que le lecteur ne soit pas surpris s'il rencontre ce terme.

4.4.5 – Jeu d'instructions orthogonal.

Cette caractéristique est plus intéressante car elle correspond à une conception simple et donc à une facilité d'utilisation des instructions. On entend par orthogonalité, l'indépendance entre les codes opération et les modes d'adressage.

Le jeu d'instructions du 6502 n'est pas orthogonal. Toutes les instructions ne bénéficient pas de l'adressage relatif, seuls les branchements conditionnels comme BNE en disposent. On pourrait multiplier le nombre de tels exemples dans le cas du 6502.

Le 68000 de Motorola a par contre un jeu d'instructions orthogonal, ce qui explique en partie son petit nombre d'instructions (deux fois moins qu'un 8086 auquel on peut le comparer). De tels processeurs sont plus faciles à utiliser pour le programmeur, mais aussi pour les compilateurs. On retrouve là un des objectifs des processeurs RISC.

Le DPS8 a aussi son jeu d'instructions initial (sans les instructions commerciales) orthogonal (figure 4k).

Comme trop souvent en informatique, le vocabulaire n'est pas clairement défini par une autorité ayant une compétence reconnue, et on parlera de symétrie ou de régularité (littérature INTEL sur l'iAPX 432 par exemple) au lieu d'orthogonalité.

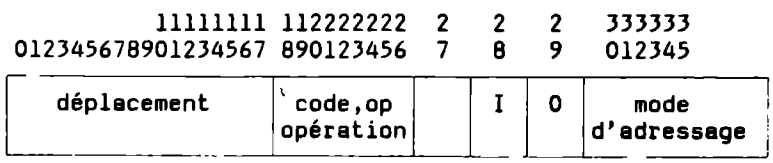


Figure 4k: instruction commerciales des DPS8.

4.4.6 – Nombre d'opérandes.

On se réfère généralement au nombre d'opérandes de la majorité des instructions pour parler de **machine à n adresses.**

Il existe des machines à **trois opérandes** comme les CYBER de CDC (cas des instructions avec adressage registre direct; ces registres doivent au préalable être chargés depuis la mémoire centrale avec des instructions à deux adresses), l'iAPX 432 d'INTEL, le VAX de DEC et les instructions commerciales (EIS) des DPS8. On trouve de telles machines dès le MIDAC, successeur du SEAC (1950). Une addition rangeant dans OP3 le résultat de la somme de OP1 et OP2 s'écrira alors:

op-code ADD	OP3	OP2	OP1

La majorité des ordinateurs actuels ne manipule pas d'aussi longues instructions. Ils ont des instructions à deux adresses, ce qui est satisfaisant dans la plupart des cas. Cette tendance s'est affirmée depuis les années 60 avec la sortie des IBM 360. Notre exemple précédent nécessitera alors deux instructions:

op-code MOVE	OP3	OP1	initialise OP3 avec OP1
op-code ADD	OP3	OP2	ajout de OP2 à OP1 dans OP3

Cet exemple montre que l'un des deux opérandes est implicitement le résultat de l'opération: il sert d'entrée au début, de sortie à la fin. Sur 360/370 d'IBM, DPS7 de Bull, c'est le premier opérande (la première adresse logique qui suit le code opération) qui recevra le résultat.

Dans les ordinateurs de bas de gamme, comme les micro-ordinateurs basés sur des microprocesseurs 8 bits, on a affaire à des machines à **une adresse.** Un des opérandes et l'adresse du résultat sont implicites. Il s'agit généralement d'un registre appelé **accumulateur.** Il faut alors 3 instructions pour réaliser notre addition:

op-code LOAD	OP1	charge OP1 dans l'accumulateur
op-code ADD	OP2	accumulateur = OP1 + OP2
op-code STORE	OP3	range l'accumulateur en OP3

Le 6502, le DPS8 dans son jeu d'instructions de base, sont des machines à une adresse.

Nous venons de voir qu'en principe là où les machines à 3 adresses n'utilisaient qu'une seule instruction, les machines à 2 adresses en utilisent deux et les machines à 1 adresse ont besoin de trois instructions.

Il est possible de présenter les instructions travaillant sur des piles de calcul comme des instructions à **zéro adresse**. L'exemple d'addition de deux mots sur un TANDEM s'écrira:

```
LOAD G+2   charge au sommet de la pile le mot d'adresse 2, et celui
LOAD G+3   d'adresse 3 de la zone réservée aux variables
IADD
STOR G     range en mémoire, à l'adresse G, le résultat
```

L'instruction IADD est une instruction à zéro adresse et l'on constate qu'il nous faut 4 instructions pour notre exemple d'addition. On trouve ce type d'instruction dans les Burroughs 5500, 6700, les HP3000, les TANDEM, les ICL2900, les iAPX432.

Les exemples ci-dessus sont quelque peu trompeurs car ils ne tiennent pas compte de l'emplacement des opérandes. S'ils sont tous dans des registres, les exemples sont fidèles à la réalité. Si ces opérandes se trouvent en mémoire centrale, il faudra en fait trois instructions sur un 360/370 au lieu de deux:

op-code LOAD	Ri	OP1	charge OP1 dans Ri
op-code ADD	Ri	OP2	Ri = OP1 + OP2
op-code STORE	Ri	OP3	range Ri dans OP3

C'est pourquoi Motorola parle de machine à 1,5 adresse dans le cas du 6809, un des opérandes est dans un registre, l'autre en mémoire centrale.

4.5 – Machines à plusieurs jeux d'instructions.

Un ordinateur a généralement le jeu d'instructions qui a été conçu pour lui ou qui est celui de la gamme à laquelle il appartient. Dans certains cas les ordinateurs ont plusieurs jeux d'instructions, un seul étant actif à un moment donné dans un processeur, mais plusieurs peuvent résider simultanément dans l'ordinateur. La micropro-

grammation que nous étudierons plus loin (chapitre 11) a énormément facilité la réalisation de ces machines à plusieurs jeux d'instructions.

Quand nous parlons de plusieurs jeux d'instructions, nous excluons le cas des ordinateurs dont le jeu d'instructions contient celui d'un autre ordinateur: par exemple le jeu d'instructionss du Z80 est un surensemble de celui du 8080.

Une des premières machines à disposer ainsi de plusieurs jeux d'instructions fut le DPS7 (appelé 64 à l'époque), conçu au début des années 70. Il avait en plus de son propre jeu d'instructions (son code natif):

- le jeu de l'IBM 360,
- celui du G100 (un mini-ordinateur de General Electric),
- celui des H200/2000, machines de gestion d'Honeywell.

Ces jeux supplémentaires appelés "décors" avaient pour but d'offrir une évolution aux clients Bull équipés de G100 et de H2000; le décor 360 avait lui pour but de reprendre des clients d'IBM. Depuis, le DPS7 a au début des années 80 repris les jeux d'instructions des IRIS 50 et IRIS 80 de l'ex-CII. Fusion oblige.

DEC est dans un cas analogue avec le VAX qui possède en plus de son jeu natif, le jeu d'instructions des PDP11. Mais DEC a renoncé, après l'avoir longtemps laissé espérer, à un nouveau VAX reprenant aussi le jeu d'instructions des PDP10.

Les Burroughs B1700, B1800, B1900 (le premier chiffre identifie la série, le second l'évolution technologique) offrent aussi trois jeux d'instructions. Nous en parlerons au paragraphe suivant.

4.6 – Machines orientées langages de haut niveau.

Nous avons traité jusqu'ici des machines qui ont un jeu d'instructions de très bas niveau et pour lesquelles les compilations lourdes (en temps et en espace mémoire occupé par les compilateurs) sont inévitables.

Un certain nombre de recherches vise à mettre au point des machines évitant ces compilations. Ceci peut être obtenu en faisant directement interpréter par le matériel les instructions d'un langage de haut niveau. On parle alors de machine-langage ou de machine orientée vers les langages de haut niveau (LHN, HLL en américain). Là encore la microprogrammation a offert une solution réaliste.

Une des premières machines de ce style fut la B1700 de Burroughs lancée au début des années 70. Elle dispose de 3 jeux d'instructions de haut niveau: le Fortran, le Cobol et un langage de type Algol utilisé par Burroughs pour écrire son système d'exploitation.

Depuis on a vu apparaître des machines orientées vers le Pascal, machines interprétant directement le P-code, des machines orientées vers le Lisp, des mchineorientées vers l'APL.

chapitre 5

rappels sur les circuits, les signaux, la technologie

La lecture de ce chapitre est facultative. Il est destiné au lecteur qui n'a aucune notion concernant les circuits électroniques, circuits que nous évoquerons dans les chapitres suivants lorsque nous détaillerons quelques aspects de l'architecture interne (registre instruction, circuit de transformation des adresses, etc.).

5.1 – Les éléments de base.

L'élément de base du sous-système central est le **transistor**. Ces transistors sont alimentés par l'énergie fournie par les alimentations (une ou plusieurs de voltages différents selon les technologies) qui ne jouent aucun rôle sur le plan logique.

Toute la vie de l'ordinateur (le déroulement des instructions) est véhiculée par des **signaux** que s'échangent les transistors. A chaque signal **l'état** interne (le contenu de tous les transistors) de l'ordinateur évolue et l'élaboration des résultats progresse; on parle aussi de **transitions** entre deux états successifs de l'ordinateur.

Si autrefois les transistors étaient manipulés isolément (on parlait alors de **composants discrets**), le logicien actuel manipule directement des ensembles de transistors que l'on appelle des **circuits intégrés** (CI, IC en américain) ou puce (chip).

5.2 – Les signaux.

Les signaux échangés entre les transistors peuvent prendre deux formes:

- **niveaux** : l'une des deux valeurs du signal représente un 1 (un) logique, l'autre un zéro, et cette valeur du signal est maintenue tant que l'information véhiculée n'est pas modifiée,
- **impulsions** : une des valeurs de l'information est représentée par un signal constant, l'autre valeur par un changement bref du signal que l'on appelle une impulsion (pulse).

On appelle signal **d'horloge** ou horloge tout court, un signal de type impulsionnel qui sert de référence comme on le verra dans les circuits de type synchrone. Il définit à quel rythme va "vivre" l'ordinateur.

Figure 5a: types de signaux échangés entre transistors.

5.3 – Les circuits.

Un ensemble de composants (discrets ou intégrés) constitue un circuit qui réalise une fonction donnée. Un circuit est représenté de façon abstraite par une "boîte noire" avec des entrées et des sorties, les signaux de sortie matérialisent une information S reliée à l'information d'entrée E par une fonction F (figure 5b) que réalise le circuit.

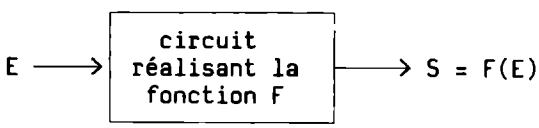
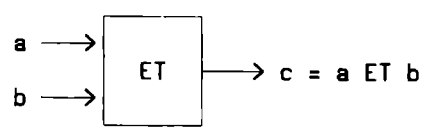


Figure 5b: représentation abstraite d'un circuit.

On distingue deux grandes catégories de circuits:

- les circuits **combinatoires** dans lesquelles la sortie S ne dépend que de l'entrée E. Les fonctions ainsi réalisées correspondent à des fonctions **booléennes**. Lorsque le circuit est simple, on parlera d'opérateur booléen (figure 5c). La réalisation de ces circuits fait appel à la synthèse des fonctions booléennes sur le plan théorique;



a, b et c sont des variables booléennes simples

Figure 5c: opérateur booléen.

- les circuits **séquentiels** où la sortie S ne dépend pas seulement de l'entrée E, mais aussi du **temps** t, par l'intermédiaire de **l'état** du circuit. Cet état du circuit est matérialisé par des transistors dont l'état évolue au cours du temps en fonction des différentes entrées E(t), donc des états précédents. Sur le plan logique on associera à ces transistors une variable (une information) **interne** I, c'est-à-dire non significative pour l'utilisateur qui lui ne s'intéresse qu'à S et E reliés par la fonction F (figure 5d).

  Le circuit séquentiel, contrairement au circuit combinatoire, réalise une fonction de **mémorisation**. Le circuit séquentiel le plus simple est une **bascule** qui permet de stocker une variable booléenne simple (un bit).

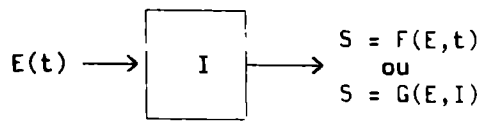


Figure 5d: circuit séquentiel.

Un circuit séquentiel complexe se ramène toujours (figure 5f) à un ensemble formé de:
* deux circuits combinatoires,
* bascules, une par variable interne simple.

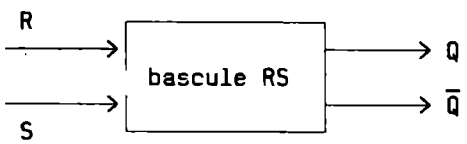


Figure 5e: un circuit séquentiel simple.

une entrée R (Reset) permet de mettre zéro dans la bascule, l'entrée S (Set) y met un. Q est en même temps la variable d'état interne et de sortie.

Un circuit séquentiel **synchrone** est un circuit où les transitions (les changements d'état) ne se font que lorsqu'un signal de référence, le signal d'horloge, produit une impulsion. Les variables d'entrée et l'état des bascules ne sont pris en compte qu'à ce moment là.

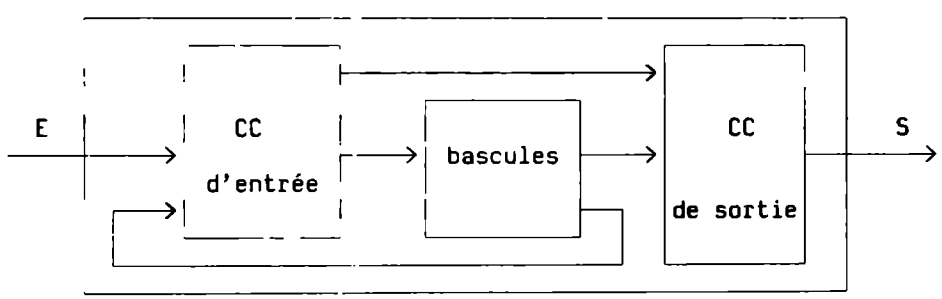


Figure 5f: structure générale d'un circuit séquentiel (CC = circuit combinatoire).

Dans le cas des circuits séquentiels **asynchrones** il n'y a pas de signal de référence et toute évolution d'une variable d'entrée produit des transitions, quelque soit l'instant où elle apparaît.

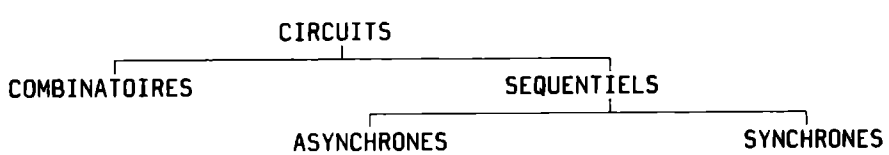


Figure 5g: typologie des circuits.

Les mémoires que nous avons vues au chapitre 3 sont des circuits séquentiels généralement synchrones lorsqu'elles sont réalisées avec des semi-conducteurs. E y est alors l'adresse, S le contenu de l'information à l'adresse E dans le cas d'une lecture (figure 5f).

5.4 – Les circuits classiques.

Nous rappelons brièvement quelques circuits très fréquemment présentés dans les ouvrages spécialisés:
- additionneur;
- circuit de multiplication;
- registre à décalage; les technologies CCD (charge coupled device) et MBM (bulles magnétiques) se prêtent naturellement à la réalisation de ces circuits. On distingue les registres à décalage **statiques** dans lesquels l'information ne se déplace que sur demande, et les registres à décalage **dynamiques** dans lesquels les bits défilent de façon continue de cellule en cellule;
- circuit de décodage;
- base de temps qui délivre les signaux d'horloge;
- multiplexeur.

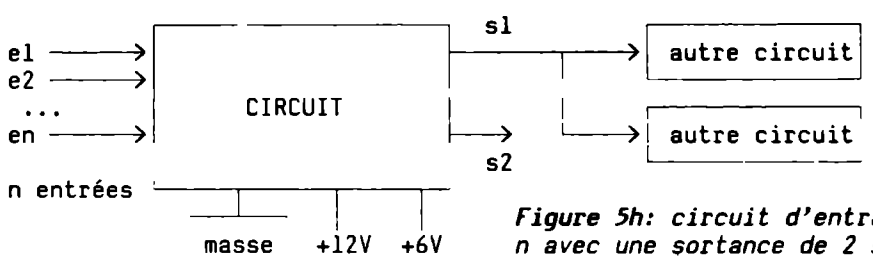


Figure 5h: circuit d'entrance n avec une sortance de 2 sur s1.

5.5 – Technologie de base.

Il existe différentes technologies basées sur les semi-conducteurs. Une technologie porte un nom (TTL, ECL, CMOS, etc.) et correspond au fait que les circuits d'une même technologie sont compatibles sur le plan physique:

1 ils peuvent être directement interconnectés entre eux car les signaux (de sortie) délivrés par un circuit ont des caractéristiques (amplitude, fréquence, puissance, etc.) telles qu'ils peuvent constituer les signaux d'entrée d'autres circuits;

2 la logistique (alimentation, masse, tensions de référence, etc.) est la même pour tous;

3 la vitesse de fonctionnement caractérisée par le temps nécessaire au changement d'état des transistors est analogue.

L'entrance (nombre maximum d'entrées qu'un circuit peut accepter, fan in en américain) et la sortance (nombre maximal d'entrées d'autres circuits qu'un signal de sortie peut "attaquer", fan out en américain) ne constituent pas des limites.

La série 3090 d'IBM est en ECL. Le 8086 est en HMOS, le 80C86 en est une version CHMOS (consommation standard de 10 milli-ampères). Une EPROM de 256 Kbits en CHMOS II offre en 1985 un temps d'accès de 170 ns. Le microprocesseur 32 bits du HP9000 est en NMOS III avec des traits de un micron de large.

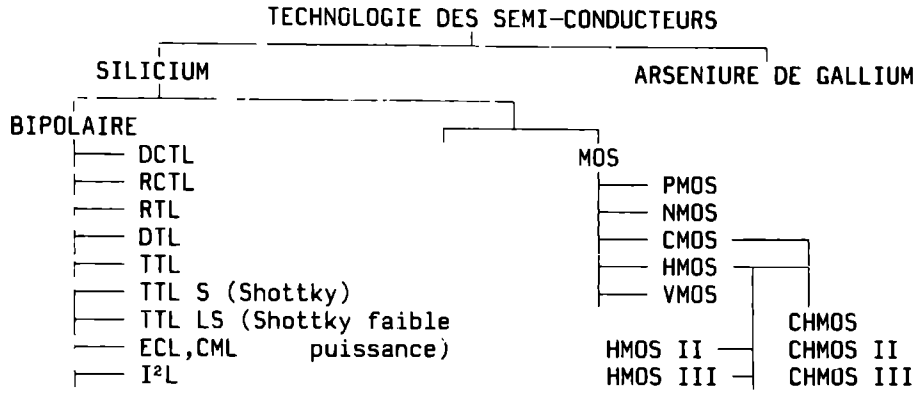


Figure 5i: classement des principales technologies.

5.6 – L'intégration.

Ce terme est propre aux circuits intégrés et correspond à la densité des transistors sur le silicium (actuellement) que l'on sait atteindre. Cette densité double tous les ans selon [MEA83].

Les grandes étapes de l'intégration sont marquées par des noms attribués aux différents niveaux d'intégration:

- SSI pour intégration à petite échelle (Small Scale Integration): on atteignait une dizaine de portes par millimètre carré de matériau;
- MSI pour intégration à moyenne échelle: on passe d'une dizaine à quelques centaines de portes par millimètre carré. Les puces ont une surface d'une dizaine de mm^2 et se présentent sous forme de DIL (ou DIP, dual in line package) rectangulaires c'est-à-dire de boîtiers de 5 cm avec deux rangées parallèles de broches (pin en américain), jusqu'à 40 typiquement;
- LSI pour intégration à grande échelle au début des années 70: environ 5000 transistors par millimètre carré. Le 4004 comportait environ 2000 transistors au début 72, le 8008 en avait 3500 fin 72, et le 8080 5000;
- VLSI pour intégration à très grande échelle: les puces peuvent atteindre 100 mm^2 et jusqu'à 200 broches. Les boîtiers se présentent sous forme de DIL, de QIL (carré) et de PCC (plastic leaded chip carrier) à 124 broches en quatre rangées sur les quatre côtés ("fakir"), et même de PGA (pin gread array) et de LCC (leadless ceramic chip carrier, 172 broches) avec les broches sous le boîtier et non plus sur les côtés.

Le 8088 comporte 29 000 transistors, le 68000 70 000, le 80286 plus de 100 000, quant à l'iAPX 432 il se présente sous forme de trois

boîtiers de 8,4 mm de côté et comprend 120 000 transistors. Pour [UBE85] la technologie actuelle permet d'atteindre 200 000 transistors, ce qui ne suffit pas à implanter sur une seule puce les fonctions d'un processeur avec gestion de mémoire virtuelle et flottant; d'où les coprocesseurs et les MMU (memory management unit) qui ne sont d'ailleurs pas nécessaires dans les petits systèmes. Affirmation dépassée en fin 85 avec l'annonce de l'iAPX 386 d'Intel qui comporte 275 000 transistors, MMU comprise. Pour donner une idée de l'évolution de cette intégration, [DEH85] indique qu'en 1960 une unité centrale comprenait environ 200 000 portes, soit un million de transistors discrets; en 1970 5 à 10 000 boîtiers à faible intégration (SSI) suffisent; en 1984 on tombe à 20 boîtiers VLSI. Le coût des circuits intégrés est divisé par deux tous les 18 mois environ. [UHL81] cite page 134 la réduction de 95% du nombre de composants entre 1973 et 1977 pour un ensemble comprenant un processeur, 2 Kb de MEM/PROM, 256 bits de MEV, 14 temporisateurs (interval timers), 5 portes d'entrées/sorties, 4 niveaux d'interruption et une entrée-sortie série.

TECHNOLOGIE	TEMPS DE TRAVERSEE (ns)	CONSOMMATION PAR PORTE (mW) **	APPARUE EN
transistor	20		1960
DTL	20		1965
TTL	10	2	1965
TTL LS	8	0,4	
TTL S	3	3,75	
I2L	> 4	> 0,16	
ECL (CML)	0,75 à 2	25 à 40	1970
PMOS	100 à 200	0,14	1975
NMOS	50 à 100	0,1	1980
CMOS	à 5V: 60	0,001	1984
	à 10V: 30	0,01	
HBMOS	2		1980

** Une porte équivaut selon les auteurs à 2 ou 3 transistors.

Tableau 5j: comparaison de quelques technologies.

Il y a deux niveaux d'intégration à distinguer dans les ordinateurs:

- le premier niveau est celui que nous venons de voir et concerne le nombre de transistors sur une puce (un CI). Ce nombre est limité par la largeur des traits qui sont "dessinés" sur le silicium afin de former les transistors et leurs interconnexions. Cette largeur est de un à quelques microns actuellement et devrait descendre en-dessous du micron dans les années qui viennent;

- le second niveau concerne l'assemblage de ces puces pour former des circuits d'ordinateur. Cet assemblage se fait selon deux grands types:
  * assemblage des boîtiers céramiques (applications professionnelles) ou plastiques (grand public) sur des cartes (plaques),
  * assemblage des circuits intégrés, sans les boîtiers, lorsque l'on vise la miniaturisation ou les performances élevées. La série 308X d'IBM est ainsi constituée de modules (TCM) contenant jusqu'à 133 puces, montés sur des circuits imprimés multicouches; le 3083 comporte 8 TCM, sans aucun câblage externe.

Si le premier niveau relève des fabricants de semi-conducteurs comme Texas Instruments, INTEL, NEC, etc, le second niveau est un problème qu'ont à résoudre les constructeurs qui achètent les puces auprès des "fondeurs" de silicium.

Dans les années 70 Gene Amdahl, le concepteur des 360/370 puis des compatibles qui portent son nom, indiquait qu'un tiers du temps qui s'écoule dans le processeur correspondait au transfert des signaux entre les puces d'une même carte, un autre tiers au temps de transfert des signaux entre cartes. On comprend donc l'intérêt des constructeurs pour avoir des ordinateurs aussi compacts que possible. C'est dans ce but qu'Amdahl fonda une nouvelle société (Trilogy) pour concevoir de nouveaux compatibles IBM basés sur des circuits occupant une tranche entière de silicium (WFI pour Wafer Scale Integration) alors qu'une telle tranche produit généralement un nombre important de puces, couramment plus d'une centaine. Mais Trilogy a dû renoncer récemment à ces deux objectifs, les compatibles IBM, et les circuits sur une tranche entière.

5.7 – Les composants que l'on peut acheter.

Le concepteur d'une carte dispose directement de composants avec un niveau de fonctionnalité élevé. Dans "le panier de la ménagère" on trouve:

- les microprocesseurs;
- des puces mémoire. Les puces à 256 Kbits sortent actuellement, les 1 Mbits (1 200 000 transistors pour des ROM) sont testées en laboratoire
- des opérateurs;
- les circuits en tranche pour les processeurs rapides;
- les circuits personnalisables (customizable), pour des séries d'au moins mille circuits;
- les circuits programmables (PAL,PLA pour programmable logic array). Ils permettent de modifier facilement la fonction réalisée par le circuit, sans en changer la conception. Ils conduisent à une représentation minimum de la fonction à réaliser sous forme de OU de monômes booléens. Pour les mémoires de commande (chapitre 11) ils offrent une solution moins coûteuse que les mémoires mortes.
- des bus;
- toute la périphérie: PPI, VIA, PIO, UART, USART, CRTC, etc.

chapitre 6

les objets logiques manipulés par les instructions

6.1 – Les objets élémentaires et les indicateurs.

Les objets les plus simples manipulés par l'ordinateur sont:

- les caractères,
- les nombres entiers positifs manipulés par les instructions arithmétiques,
- les chaînes de bits manipulées par les instructions logiques.

C'est ce que l'on trouve dans les microprocesseurs 8 bits comme le 6502 où il n'y a pas de différence réelle entre ces types de données: le caractère, le mot, la chaîne de bits ont tous 8 bits.

Il est cependant possible dans ces machines de manipuler des nombres entiers négatifs grâce aux **indicateurs**. Les indicateurs sont des bits ou des groupes de bits positionnés par les instructions de traitement en particulier:

- l'indicateur Z (comme Zéro) indique si le dernier calcul (addition par ADC sur 6502 par exemple) a produit un résultat nul (Z mis à un) ou non (Z mis à zéro). L'instruction LDA sur 6502 a comme code opération A9 en hexadécimal et occupe deux octets pour ce code opération. Dans l'exemple ci-dessous:

A9	00

LDA charge l'accumulateur (un des registres du processeur, celui dans lequel on range les résultats dans un 6502) avec la valeur prise dans le second octet de l'instruction

- l'indicateur N (comme Négatif) indique si le bit de poids fort (le bit de gauche ici) du dernier résultat est égal à un ou à zéro. Un bit de gauche est interprété comme la marque d'un nombre négatif dans la plupart des ordinateurs, d'où le nom de cet indicateur. Reprenons l'exemple du 6502 avec LDA:

A9	00	met 1 dans Z et 0 dans N

A9	80	met 0 dans Z et 1 dans N puisque le mot chargé, 80 en hexadécimal, a son bit de gauche à 1.

- l'indicateur C (comme Carry, retenue ou report en français) indique si la dernière opération réalisée a produit une retenue. Si oui C vaudra 1, sinon il vaudra 0. Cet indicateur permet, entre autres, de réaliser des opérations sur des nombres codés sur 8, 16, 24, 8n bits dans les microprocesseurs analogues au 6502 (Z80, 8080, etc.). Il joue aussi le rôle de neuvième bit dans les opérations de décalage.

Toutes les instructions ne positionnent pas tous ces indicateurs. Ainsi l'instruction de chargement de l'accumulateur du 6502 (LDA vue ci-dessus) ne positionne pas C et le laisse inchangé. Le résultat d'une opération peut donc être conservé pendant plusieurs instructions. Il est logique que LDA ne positionne pas C puisque le chargement ne peut pas provoquer de retenue. L'exemple ci-dessous:

A9	FF	charge FF (base 16) dans l'accumulateur
69	01	ajoute à l'accumulateur l'opérande (1) et la retenue C

et met à 1 l'indicateur C, et met à 0 l'indicateur N puisque le résultat est 00 ou 01. Si l'on pose l'addition de FF et 01 en base 2, on obtient:

```
  1 1 1 1  1 1 1 1      qui correspond à FF
  0 0 0 0  0 0 0 1      qui correspond à 01
-------------------
1 0 0 0 0  0 0 0 0      résultat qui nécessite 9 bits
```

Le neuvième bit, égal à un, va dans C. C peut donc être considéré comme une extension de l'accumulateur pour ce type d'opération. On peut aussi considérer ADC comme une instruction à trois opérandes, deux étant implicites. Si l'on veut additionner le contenu de l'accumulateur et l'opérande, et eux seuls, il faut auparavant mettre zéro dans C, ce que montre l'exemple ci-dessous:

18		instruction CLC qui remet 0 dans C
A9	FF	charge FF (base 16) dans l'accumulateur
69	01	ajoute 1 à l'accumulateur

Le résultat rangé dans l'accumulateur vaut zéro et Z est égal à 1.

Un quatrième indicateur est utilisé dans les microprocesseurs de la génération du 6502 afin de réaliser des opérations sur des entiers signés: l'indicateur O (comme Overflow, dépassement de capacité). Cet indicateur est mis à un dès que le bit de gauche change de valeur du fait d'un résultat en dehors du segment [-128,+127]. Il joue donc le rôle de report pour les 7 bits de droite, alors que C le joue pour les 8 bits.

Sur d'autres ordinateurs on ne parlera pas d'indicateurs, mais de **code condition** (2 bits pour la série 360/370).

Le dépassement de capacité en arithmétique entière ne conduit pas toujours au positionnement d'un indicateur:
- il peut ne pas être détecté et seuls les n bits de poids faible (dans un ordinateur à n bits) du résultat seront conservés,
- une interruption peut être générée (cas des VAX, des 370) si elle n'a pas été masquée auparavant (voir Tome III).

6.2 – Les nombres entiers en binaire.

Nous ne détaillerons pas plus les nombres entiers déjà discutés dans les chapitres précédents. Le tableau 6a donne la taille des entiers sur quelques ordinateurs.

ORDINATEUR	MOT	DEMI MOT	4 BITS	DOUBLE MOT	QUADRUPLE MOT	SIGNE	NON SIGNE
6502	8	non	non	non	non	possible	oui
8086	16	8	non	non	non	oui	oui
DPS7	32	16	oui			oui	oui
DPS8	36	18	non	72	non	oui	oui
VAX	32	16					

Tableau 6a: entiers sur différentes machines.

6.3 – Les nombres flottants.

Ces nombres sont considérés comme la caractéristique du calcul scientifique et donc des ordinateurs scientifiques. Les ordinateurs simples, comme ceux basés sur le microprocesseur 6502 ne connaissent pas les flottants. On commence à les trouver sur les microprocesseurs 16 bits, comme le 8086, mais ils sont alors implémentés par des matériels externes au microprocesseur: on parlera du coprocesseur 8087 qui, associé au 8086, permet d'avoir un ensemble manipulant aussi bien les entiers (dans le 8086 et dans le 8087) que les flottants (dans le 8087). Lorsque l'architecture externe englobe la représentation flottante, différentes situations peuvent se présenter au niveau de l'architecture interne:
- la manipulation des nombres flottants est en fait réalisée par des sous-programmes (par du logiciel et non pas par du matériel). C'est le cas des 8086/8088 sans 8087. INTEL indique que le 8087 peut accé-

lérer par un facteur de 100 les traitements en flottant;
- à défaut d'une option "câblée", c'est-à-dire de circuits spécialisés, la manipulation des nombres flottants est réalisée par des microprogrammes. C'est le cas du VAX-11/780.

Le tableau 6b quelques exemples de représentation de nombres flottants dans différents ordinateurs. Les formats G et H (exposants à 11 bits) ne figuraient pas dans l'architecture initiale du VAX.

PROCESSEUR	SIMPLE PRECISION		DOUBLE PRECISION		QUADRUPLE PRECISION		INTERNE		BASE DE L'EXPOSANT
	E	M	E	M	E	M	E	M	
8087	8	24	11	53	non		15	65	2
IBM 370	8	24	8	56	16	112			16
VAX	8	24	8	56					2
			11	53					
DPS7	8	24	8	56					16
DPS8	8	28	8	64					2 ou 16

Tableau 6b: représentation des nombres flottants
(E : nombre de bits d'exposant; M : nombre de bits de mantisse).

Le format interne du 8087 permet de représenter (dans un de ses 8 registres de 80 bits) des nombres allant de 10^{-4932} à 10^{+4932}.

Les ordinateurs n'ont pas tous le même comportement face aux problèmes d'arrondi ou de troncature lors des calculs en flottant. Sur IBM 370 il y a troncature systématique. Sur DPS7 on positionne au niveau de chaque instruction un bit qui indique que l'on veut tronquer ou bien arrondir le résultat. Sur DPS8 l'instruction FRD permet d'arrondir. L'iAPX 432 offre quatre modes d'arrondi. L'arrondi est possible du fait de bits de garde que peuvent comporter les processeurs. Ce sont des bits supplémentaires à l'intérieur du processeur central (ou de l'opérateur flottant, du coprocesseur sur les microprocesseurs) qui sont perdus lorsque l'on range un résultat en mémoire centrale. Ainsi les MV de Data General offrent 4 ou 8 bits de garde (selon le bit 8 du registre FSR): avec 4 bits de garde le processeur tronque, avec 8 bits de garde il réalise un arrondi "non biaisé" (unbiased):
- si ces 8 bits ont une valeur comprise (en hexadécimal) entre 00 et 7F, le résultat n'est pas modifié,
- s'ils sont égaux à 80 on ajoute au résultat le dernier bit de garde,
- si leur valeur est comprise entre 81 et FF, on ajoute un au résultat.

Le 8087 offre aussi quatre modes d'arrondi, le mode étant défini par les deux bits RC du mot d'état: arrondi au plus proche, arrondi vers moins l'infini, arrondi vers plus l'infini, "chop" vers zéro.

Deux ordinateurs d'architectures externes différentes peuvent produire des résultats flottants différents. Un utilisateur d'IBM 370 pourra préférer ses résultats avec troncation (il y est habitué) aux résultats obtenus avec l'arrondi d'un DPS7.

6.4 – Nombres en représentation décimale.

Si les nombres flottants sont la caractéristique des ordinateurs scientifiques, la représentation décimale est typique, avec le traitement des chaînes de caractères, des ordinateurs de gestion. Les très gros calculateurs scientifiques sont d'ailleurs rarement dotés des représentations décimales et même des chaînes de caractères.

Dans le scientifique il y a beaucoup de calculs et peu d'entrées-sorties; on a donc intérêt à utiliser la représentation qui se prête le mieux aux calculs, c'est-à-dire le flottant. Dans les ordinateurs de gestion il y a beaucoup d'entrées/sorties, d'éditions d'états, pour relativement peu de calcul. On privilégie donc le format adapté aux utilisateurs, c'est-à-dire le décimal, mode dans lequel l'ordinateur conserve séparément chaque chiffre de la base 10.

Les microprocesseurs 8 bits comme le 6502 possèdent une forme restreinte de décimal, un décimal condensé (appelé aussi BCD) limité à deux chiffres, soit 8 bits (héritage du premier microprocesseur, le 4004, qui gérait avec des chiffres hexadécimaux l'affichage sur des calculettes ou des caisses enregistreuses). A un moment donné, le 6502 travaille soit en mode binaire, soit en mode décimal. Il n'a pas comme les processeurs plus évolués de codes opération différents pour les deux modes de représentation. C'est le rôle d'un autre indicateur, l'indicateur D (comme Décimal) de préciser dans quel mode se trouve le 6502. Cet indicateur est mis en mode décimal par l'instruction SED (SEt Decimal indicator), il est mis en mode binaire par CLD (CLear D).

Le Z80 qui traite aussi des décimaux condensés sur un octet utilise un mécanisme différent. L'opération se fait comme elle se fait normalement en mode binaire, mais on exécute ensuite une instruction "d'ajustemnt" DAA (Decimal Adjust A, A étant le nom de l'accumulateur). Le 8086 suit la même philosophie, mais avec deux instructions (DAA et DAS) selon que l'on a fait une addition ou une soustraction.

Le 8086 traite aussi une forme restreinte de décimal étendu, les nombres étant limités à un seul chiffre. La philosophie est identique à celle du condensé et les instructions AAA et AAS permettent d'ajuster le résultat. Les multiplications et les divisions sont même permises avec AAM (division d'un condensé par un étendu) et AAD (produit de deux étendus et résultat en condensé) pour l'ajustement décimal.

Notons que le coprocesseur 8087 n'est pas limité aux nombres flottants puisqu'il permet aussi de traiter des entiers de 16, 32 ou 64 bits, ainsi que des décimaux condensés de 18 chiffres qu'il convertit en flottant lorsqu'il les charge dans ses registres.

Les nombres décimaux ne font pas toujours partie de l'architecture externe initiale. Sur les PDP11 de DEC ils sont ainsi apparus très tard, au début des années 80 avec le PDP11-24. De même les ancêtres des DPS8, les GE600 n'avaient pas le décimal; il n'est apparu que sur les H6000.

6.5 – Chaînes de caractères.

Autres éléments caractéristiques des ordinateurs de gestion, les chaînes de caractères sont inconnues de la plupart des microprocesseurs 8 bits (cas du 6502). Le Z80 permet cependant avec ses instructions LDI et LDR de déplacer jusqu'à 64 Ko. Les microprocesseurs 16 bits n'en disposent pas toujours (cas du 68000 de Motorola).

Sur les IBM 360 les chaînes de caractères étaient limitées à 256 octets. Le 370 a introduit des "chaînes longues" et les instructions associées.

Certains ordinateurs disposent d'instructions spécialisées dans le traitement des chaînes de bits.

6.6 – Pointeurs et descripteurs.

Nous n'avons pas jusqu'ici précisé la signification des objets manipulés par les instructions. Etant donné la structure physique du sous-système central avec ses registres, sa mémoire centrale formée d'un certain nombre de cases, ce peut être une **valeur**, mais ce peut être aussi l'**adresse** d'une case dans laquelle est rangée cette valeur.

Une adresse stockée dans un registre ou dans la mémoire centrale, est appelée **pointeur**, pour bien la différencier des autres données. Lorsqu'à ce pointeur on associe d'autres caractéristiques de la donnée (sa taille, son type, ses droits d'accès, etc.) on parlera de **descripteur**.

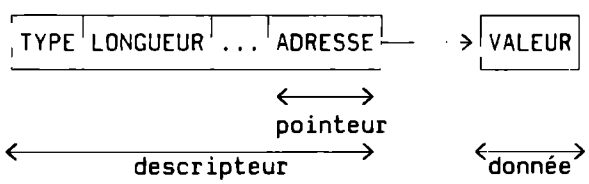


Figure 6c: relations entre valeur, pointeur et descripteur d'une donnée.

Certains ordinateurs comme les 360/370 d'IBM ne distinguent pas les pointeurs des valeurs, ils sont stockés de façon identique, et dans les mêmes registres.

Dans le cas des DPS7 ou DPS8 les représentations sont différentes. Nous détaillerons ce point lorsque nous parlerons des modes d'adressage.

6.7 – Tableaux, indices, déplacements.

Nous avons jusqu'ici parlé de données scalaires. La plupart des ordinateurs manipule les tableaux en traitant successivement chacun de leurs éléments, avec toutes les instructions de boucle, tous les tests nécessaires, pour détecter la fin du tableau.

La manipulation d'un tableau fait apparaître des numéros d'entrée dans le tableau: seul le tableau a un nom et lorsque l'on veut parler de tel élément il faut préciser son numéro.

Ces numéros sont appelés **indices** et donnent lieu à l'utilisation de **déplacements** dans la mémoire centrale. Tous les éléments d'un tableau sont rangés de façon consécutive en mémoire centrale. Soit ADR

l'adresse d'un tableau TAB, donc de son premier élément. Si chaque élément du tableau occupe une case mémoire, l'adresse de l'élément numéro i sera

ADR + i - 1

(&TAB[i-1] en C). Si par contre chacun de ces éléments occupe n cases, l'adresse de l'élément numéro i sera

ADR + n * (i - 1)

et l'on voit bien la différence entre:
- i l'indice, et
- n * (i - 1), le déplacement par rapport au début du tableau.

Les déplacements sont, comme les adresses, des objets rendus nécessaires par la structure de l'ordinateur et non par les besoins de l'utilisateur.

Il existe quelques ordinateurs, généralement orientés vers le calcul scientifique, avec des instructions capables de manipuler comme un tout des tableaux à une seule dimension (appelés vecteurs). Ainsi l'addition de deux vecteurs V1 et V2 avec résultat dans V1 s'y écrira:

ADD V1,V2

Cette instruction unique déclenchera l'addition de tous les éléments des vecteurs:

V1(i) = V1(i) + V2(i) pour i=1,2,3,...

Ces ordinateurs sont dits **vectoriels**, ou **processeurs de tableaux** (array processors). Nous en reparlerons lorsque nous aborderons le parallélisme et l'anticipation (pipelining).

6.8 – Listes.

Une liste est une structure qui peut avoir des formes différentes selon les ordinateurs qui en disposent. Dans le VAX les listes sont circulaires, avec double chaînage (avant et arrière). Une liste y est un ensemble d'éléments (d'entrées), chaque élément contenant l'adresse de l'élément suivant (chaînage avant) et l'adresse de l'élément précédent (chaînage arrière). Le VAX permet d'insérer de nouveaux éléments (instruction INSQUE), d'en retirer (REMQUE).

6.9 – Sémaphores.

Un sémaphore simple est un compteur imaginé par Dijkstra pour résoudre les problèmes de communication et de synchronisation entre processus (nous dirons "programmes" pour l'instant).

Ce compteur est mis à zéro au départ pour représenter le fait que l'événement associé ne s'est pas produit. Lorsque l'événement se produit (exécution d'une instruction appelée V, fin d'une E/S sur un DPS7 par exemple) le compteur est incrémenté. Quand un programme a besoin d'attendre que l'événement se produise avant de poursuivre son exécu-

tion, il lit le sémaphore (instruction P). Le compteur est alors décrémenté. Si sa nouvelle valeur est négative, l'événement associé ne s'est pas encore produit et le programme est mis en attente de l'événement. Si sa nouvelle valeur est positive ou nulle, l'événement s'est produit et le programme continue normalement.

Un sémaphore est donc un objet vis-à-vis duquel les programmes se conduisent comme producteurs d'événements ou comme consommateurs de ces mêmes événements.

Un sémaphore plus complexe gère une liste de programmes (processus) en attente de l'événement associé. Le DPS7 dispose de tels sémaphores au niveau du matériel. Dans certains systèmes, les sémaphores sont simulés par le logiciel de base (cas du système d'exploitation iRMX86 d'INTEL sur 8086). Dans d'autres systèmes on appelle sémaphore de simples cases mémoire manipulées par des instructions "test and set" (cas du CORALIS-80 de CSEE).

6.10 – Pages, segments.

Les pages et segments seront étudiés dans le Tome II. Ils sont en effet liés au problème de gestion des mémoires centrales et virtuelles, fonction assurée par le système d'exploitation. Ils sont d'une façon générale utilisés pour:

- rendre les programmes indépendants de leur emplacement en mémoire centrale,
- les protéger contre les accès non voulus ou non désirés,
- les rendre indépendants de la taille mémoire,
- gérer la mémoire avec des objets de taille fixe.

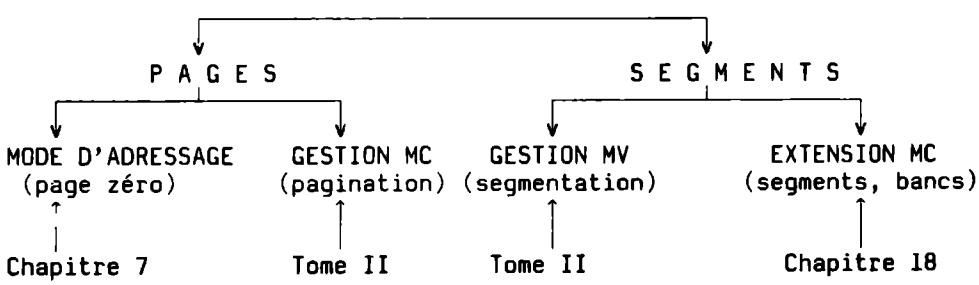


Figure 6g: traitement des pages et segments dans l'ouvrage.

Il ne faut pas confondre ces pages-là avec les pages que nous verrons lorsque nous parlerons du mode d'adressage page zéro dans le chapitre 7, ce sont deux choses totalement différentes. La page de l'adressage page zéro est liée au problème général de l'adressage, la page du Tome II est liée à la gestion de la mémoire centrale et est transparente pour l'utilisateur "normal" qui se situe au niveau de l'architecture externe.

Les segments sont eux liés à la gestion des données, et nous les étudierons dans le Tome II, dans le chapitre consacré à la gestion de la mémoire virtuelle. Les segments tels qu'on les trouve dans le 8086

par exemple seront présentés dans le chapitre 18, dans la section consacrée à l'augmentation de l'espace de travail des programmes.

La présence de ces objets dans le processeur crée des registres et des instructions particulières pour les manipuler (LDS, LES sur 8086).

6.11 – Processus.

Sur les ordinateurs comme les 370 d'IBM, les DPS8 de Bull, la grande majorité des ordinateurs, les processus (process en américain; on dit aussi tâche, task, processeur virtuel) sont des créations du système d'exploitation. Les processus correspondent à des ensembles d'instructions (de programmes) plus un contexte, un état; ils sont en concurrence entre eux pour disposer des ressources de l'ordinateur, du processeur en particulier.

Dans les DPS7, les VAX, les PRIME, les iAPX 432, les iAPX 386, les iAPX 286, etc. les processus sont par contre connus du matériel, donc connus au niveau de l'architecture externe du sous-système central.

6.12 – Machines orientées objet.

Les derniers exemples ont montré que le nombre d'objets connus de l'utilisateur peut devenir grand, que les objets peuvent être très différents les uns des autres: il n'y a pas grand rapport entre un entier et un processus.

On parlera d'architecture orientée objet lorsque les objets connus dans l'architecture externe sont clairement définis, chacun avec ses caractéristiques (son type). Les instructions seront partitionnées selon les objets sur lesquelles elles s'appliquent et il ne sera pas question de manipuler un objet de type X avec une instruction prévue pour manipuler un objet de type Y. Le matériel fera les contrôles nécessaires pour s'assurer que l'objet est bien celui prévu par l'instruction.

Nous nous trouvons aux antipodes des processeurs RISC dans la mesure où l'on obtient ainsi des processeurs complexes.

Des machines de ce type sont l'iAPX432, l'IBM 38. Nous verrons dans le Tome II une orientation identique dans certains systèmes d'exploitation orientés objets. L'orientation objet, le "typage" des objets ne se retrouve pas seulement dans les architectures externes ou les systèmes d'exploitation, elle est courante dans le domaine des langages de haut niveau comme ADA ou Pascal.

chapitre 7

les objets physiques manipulés par les instructions

Nous avons vu dans le chapitre précédent les principaux objets logiques manipulés par l'instruction. Où se trouvent physiquement ces objets? Dans la mémoire centrale, et dans des registres si le processeur en dispose (figure 7a).

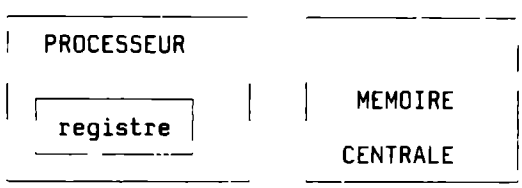


Figure 7a: structure interne du sous-système central faisant apparaître la mémoire centrale, les registres et le processeur central.

7.1 – Rôle des registres.

Les registres contiennent un sous-ensemble (généralement très restreint) des données stockées en mémoire centrale. Ils correspondent donc à une **duplication** de l'information. Cette duplication, gérée par les programmes qu'exécute le processeur, se justifie par des problèmes de performance: le temps qu'il faut au processeur pour obtenir les services de la mémoire centrale lui paraît relativement long, aussi lorsqu'il veut travailler sur des données, il préfère les amener au préalable dans ses registres. Le rapport des temps d'accès de la mémoire centrale et des registres est fréquemment supérieur à 10 (quelques dizaines).

On assiste là à l'apparition d'une première **hiérarchie des mémoires**. Plus nous avancerons dans ce tome, plus cette hiérarchie s'étoffera et on constatera que cette évolution se fera en suivant un phénomène constant: plus on s'éloigne du processeur central, plus les mémoires deviennent volumineuses, et en même temps plus lentes.

On peut constater ce rapport des volumes dans le cas du 6502: face aux 64 Ko que peut atteindre la mémoire centrale, les registres n'offrent guère que 8 bits (ceux de l'accumulateur), 24 si l'on espère stocker des données dans les deux registres X et Y. Sur un 308X d'IBM (successeur des 370) qui peut atteindre plusieurs méga-octets (32 Mo)

de mémoire centrale, on trouve 16 registres généraux de 32 bits, 4 registres scientifiques de 64 bits, soit en tout 768 bits ou 96 octets. Et encore tous ne contiennent généralement pas des données.

Mais le rôle des registres ne se borne pas à cette accélération des temps d'accès. Il y a aussi de nombreux registres utilisés par le processeur pour ses propres besoins. C'est ainsi que les indicateurs sont très fréquemment regroupés dans un même registre (que nous appellerons IR comme Indicateurs Registre). L'instruction en cours d'exécution est elle aussi stockée dans un registre (appelé RI comme Registre Instruction dans ce tome), etc.

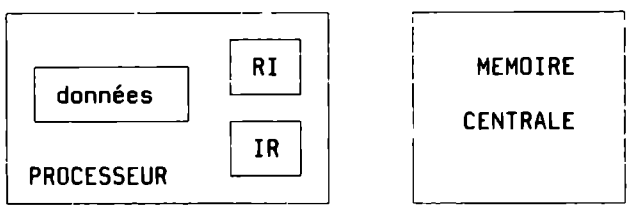


Figure 7b: registres dédiés principalement aux données et quelques registres utilisés par le processeur pour se gérer.

7.2 – Les registres réservés aux données.

Dans les processeurs ayant de tels registres (c'est-à-dire la grande majorité d'entre eux à l'heure actuelle à part des machines comme l'iAPX 432, l'IBM38, le HP9000) on parlera **d'accumulateurs** pour indiquer que ces registres contiennent des données qui seront les opérandes des opérateurs de calcul. Ce type de registre recevra de plus les résultats élaborés par ces opérateurs de calcul (figure 7c).

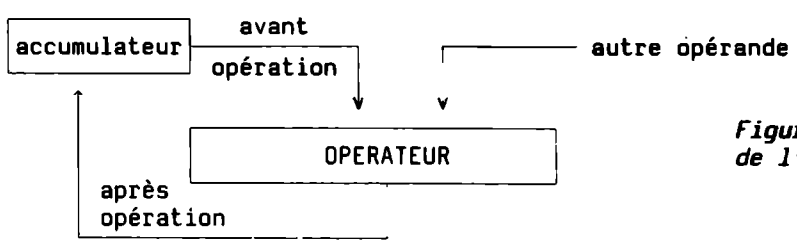


Figure 7c: rôle de l'accumulateur.

Le 6502 dispose d'un accumulateur appelé A et déjà utilisé plus haut par l'instruction ADC comme entrée de l'additionneur au début de l'opération, puis comme zone de rangement du résultat en fin d'opération.

Le 8080 a, lui aussi, un accumulateur; le Z80 en possède deux (A et A'); dans le 8086 l'accumulateur a 16 bits (AX). Le DPS8 a lui aussi un seul accumulateur de 36 bits, deux avec l'extension Q de l'accumulateur A qui lui permet de réaliser également des opérations sur 72 bits.

Les IBM 360/370 ont 16 accumulateurs de 32 bits qui tous peuvent être source ou destination des opérations. Comme ils peuvent jouer d'autres rôles que celui d'accumulateur, ils sont appelés **registres généraux**.

Dans la plupart des processeurs, les accumulateurs utilisés pour les calculs en représentation flottante sont physiquement distincts de ceux utilisés pour les calculs en entier. L'IBM 370 dispose ainsi de 4 accumulateurs de 64 bits pour le flottant, appelés **registres scientifiques**. Le coprocesseur 8087 d'INTEL dispose de 8 registres de 80 bits. Dans le DPS8 l'accumulateur est le même, en entier et en flottant.

7.3 – Registres contenant les adresses.

Dans le cas des 360/370 dont nous avons dit que l'architecture externe ne différenciait pas les données des adresses au niveau du stockage, ce sont les 16 mêmes registres généraux (d'où leur nom) qui contiennent aussi les adresses dont on a besoin (l'instruction LA comme Load Address permet de charger l'adresse d'une donnée).

Dans les DPS7 par contre, où l'on fait une distinction très nette entre adresse et valeur, le processeur dispose de 16 registres de 32 bits pour les calculs entiers, et de 8 registres supplémentaires pour les adresses.

Le 68000 de Motorola dispose aussi de 8 registres de 32 bits pour les données (Di) et de 8 autres registres pour les adresses (Ai).

Dans le 8086 on trouve aussi des registres dont la fonction est associée aux adresses: le registre de base BX, les pointeurs de base BP et de pile SP, les quatre registres de base des segments CS, DS, SS et ES. Le 8086 est une machine à registres spécialisés.

7.4 – Registres contenant les déplacements.

Nous avons introduit la notion de déplacement lorsque nous avons abordé les tableaux dans le précédent chapitre. Sur IBM 360/370 ce sont les 16 registres généraux qui peuvent contenir ces déplacements.

Sur DPS7 ce sont les registres accumulateurs qui peuvent contenir des déplacements, ce qui est assez logique puisqu'un déplacement n'est pas porteur de la notion d'adresse.

Sur le 68000 de Motorola les 16 registres peuvent contenir des déplacements. Les registres associés aux déplacements sont souvent appelés **registres d'index** et le 6502 en contient deux appelés X et Y. Sur 8086 on rangera dans cette catégorie les registres "source index" SI et "destination index" DI.

7.5 – Classification des registres.

Les registres que nous venons de voir sont bien évidemment tous visibles des instructions. On dit qu'ils sont **adressables** par ces instructions et un bon exemple est celui des instructions de format RR (Register Register) des 360/370 d'IBM donné figure 7d.

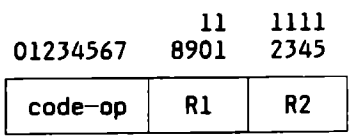


Figure 7d: instructions 360/370 de format RR.

On constate figure 7d que les deux registres manipulés par l'instruction sont définis chacun par un champ de 4 bits appelé R1 et R2. Quatre bits permettent bien de définir 16 registres différents. On dira que R1 et R2 sont les numéros (ou les adresses de façon quelque peu abusive a priori).

Il en est de même pour l'instruction MOV du 8086 qui permet de manipuler deux registres (figure 7e).

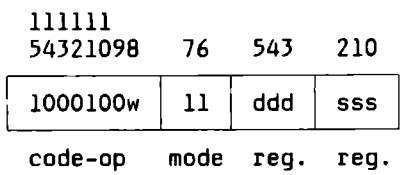


Figure 7e: format de l'instrucion MOV entre 2 registres dans le 8086.

On constate figure 7e que le 8086 permet "d'adresser" 8 registres puisqu'il code leur numéro sur 3 bits. Ce sont les registres accumulateur, déplacement et adresse (à l'exclusion des registres contenant les adresses de base des segments) et l'on peut en conclure que dans la pratique le matériel n'offre aucun mécanisme interdisant le mélange des valeurs des données et de leurs adresses.

Lorsque l'on parle d'ordinateur sans registre, il faut comprendre sans registre données adressable par les instructions.

En dehors des registres adressables il y a des registres **non adressables** dont certains peuvent rester inconnus du programmeur. Ainsi le registre RI contenant l'instruction en cours d'exécution. Un autre exemple concerne le 8008 d'INTEL où l'instruction

MOV D,S (codée 11dddsss en binaire)

se déroule en deux parties (on dira deux cycles): le contenu de S est d'abord transféré dans un registre **transparent** pour le programmeur, durant la seconde partie le contenu de ce registre non adressable est transféré dans le registre destination D. Il en va de même des registres descripteurs de segments sur l'iAPX 286.

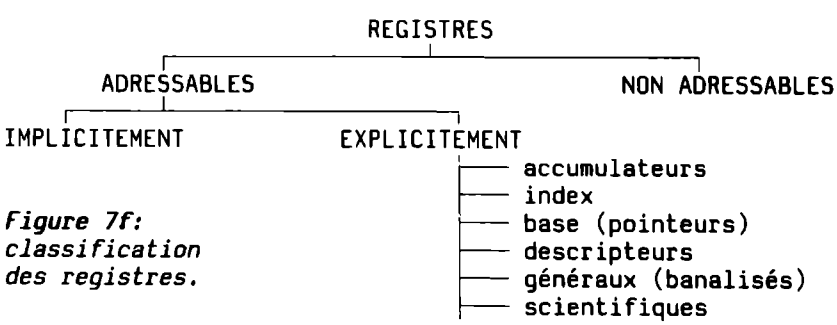


Figure 7f: classification des registres.

7.6 – Registre instruction et compteur ordinal.

Le processeur central contient au moins deux registres dédiés à l'enchaînement des instructions. Nous avons déjà vu le registre ins-

truction RI qui contient l'instruction en cours d'exécution, celle sur laquelle travaille le processeur. Un second registre appelé compteur ordinal, abrégé en CO (program counter, PC, en américain), contient l'adresse de la prochaine instruction à exécuter.

Comme le processeur suppose que les instructions sont rangées en séquence en mémoire centrale, il ajoute systématiquement la longueur de l'instruction en cours d'exécution au compteur ordinal comme le montre la figure 7g.

RI	CO	
XXXXXX	2A43	avant exécution de CLC (codé 18)
18XXXX	2A44	après exécution de CLC
6D813C	2A47	après exécution de l'ADC (codée 6D)

Figure 7g: évolution de RI et CO du fait de l'exécution des deux instructions CLC et ADC (XX représente un octet de contenu inconnu).

Seules les instructions de "rupture de séquence" comme les branchements, les appels de sous-programmes peuvent modifier ce mécanisme élémentaire. L'exemple de la figure 7h montre l'effet d'une instruction JMP, toujours dans le 6502.

RI	CO	
65A7XX	20B9	avant exécution de JMP
4C4330	3043	après exécution de JMP

Figure 7h: effet d'une instruction de branchement sur le compteur ordinal.

On y constate qu'une telle instruction "écrase" le contenu précédent du compteur ordinal pour y "forcer" une nouvelle valeur. La figure 7h montre encore que dans le 6502 les deux octets du champ adresse d'une instruction comme JMP sont inversés: le code (4A) de JMP est suivi de l'octet de poids faible (43) et non pas de l'octet de poids fort (30), ce qui explique que le compteur ordinal contient 3043, les poids forts étant à gauche, mode plus naturel puisque conforme à notre façon de faire en base dix.

Nous en arrivons ainsi à un processeur central avec deux sous-ensembles bien typés: l'un est consacré aux données et contient les registres accumulateurs avec les opérateurs, l'autre est spécialisé dans la manipulation des instructions avec RI, CO et généralement un opérateur dédié à l'incrémentation du compteur ordinal (Figure 7i).

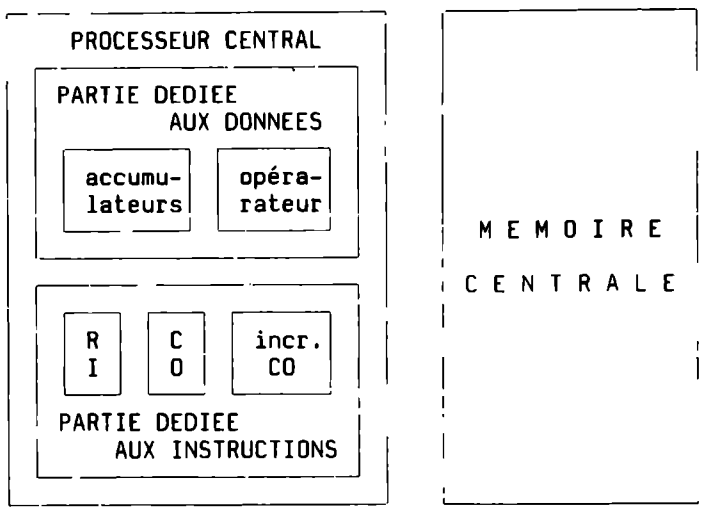


Figure 7i: structure actualisée du sous-système central.

7.7 – Les modes d'adressage.

Nous avons indiqué précédemment que l'instruction ne contenait pas obligatoirement la valeur de chacun de ses opérandes, mais leur adresse. Une telle adresse est le numéro d'une case de la mémoire centrale. Pour différentes raisons, l'instruction ne contient pas toujours directement ce numéro:

- plus la mémoire est grande, plus ce numéro occupera un grand nombre de bits (16 pour les 64 Ko du 6502, 20 pour les 1 Mo du 8086, 24 pour les 16 Mo d'un 370, etc.). Or un programme travaille à un moment donné dans une (ou plusieurs) petites zones de la mémoire, il n'est donc pas utile de donner à chaque instruction la possibilité d'atteindre rapidement toutes les cases mémoire. Les modes d'adressage auront par conséquent comme objectif de **réduire la taille** des instructions;
- il est souvent intéressant pour obtenir plus de souplesse et plus de généralité (dans la gestion de la mémoire centrale, dans l'appel de sous-programme avec paramètres), de rendre les instructions indépendantes de l'adresse où sont stockées les données. Les modes d'adressage ont donc aussi pour objectif de rendre instructions et données **indépendants** de leur adresse de stockage en mémoire centrale;
- dans les ordinateurs servant simultanément plusieurs utilisateurs il est indispensable de les protéger les uns des autres, et de protéger le système d'exploitation. Certains modes d'adressage permettent ainsi d'augmenter la **sûreté** et la **sécurité** de fonctionnement de l'ordinateur.

Les modes d'adressage, relativement nombreux, sont de plus sujets à une grande variété d'appellations. Chaque constructeur a tendance à inventer des termes qui lui sont propres. Nous nous efforcerons de définir une terminologie unique et aussi cohérente que possible.

7.8 – Adresses logiques, physiques, effectives.

Avant d'aborder les différents modes d'adressage il nous faut définir un certain nombre de types d'adresse:

- l'adresse **logique** (abrégée en AL) est celle qui est contenue dans les instructions, celle qui y définit chacun des opérandes,
- l'adresse **physique** (abrégée en AP) est celle qui est envoyée à la mémoire centrale par le processeur central,
- l'adresse **effective** est l'adresse physique finale à laquelle est stockée la valeur de l'opérande défini par l'adresse logique. Nous verrons que dans certains cas (adressage indirect par exemple) le processeur envoie plusieurs adresses physiques à la mémoire avant d'obtenir la valeur de l'opérande,
- l'adresse **réelle** interviendra lorsque nous aborderons les mémoires avec des mécanismes d'adressage non linéaires (cas des mémoires virtuelles en particulier) dans le Tome II.

7.9 – Circuit de transformation d'adresse.

Le processeur central contient un certain nombre de circuits: nous avons déjà vu les opérateurs arithmétiques et logiques pour traiter les données, le circuit d'incrémentation du compteur ordinal. Les modes d'adressage font intervenir un nouveau circuit appelé circuit de transformation d'adresse.

Comme le montre la figure 7j, le circuit de transformation d'adresse reçoit en entrée l'adresse logique AL et fournit en sortie une ou plusieurs adresses physiques AP jusqu'à fournir l'adresse effective, c'est-à-dire l'adresse physique à laquelle se trouve la donnée cherchée.

7.10 – Adressages registre et mémoire centrale.

Alors que le problème d'adressage concerne principalement la mémoire centrale, les micro-processeurs ont popularisé le terme d'adressage registre.

Nous débuterons cependant par les modes d'adressage de la mémoire centrale et présenterons ensuite les modes d'adressage registre dont certains font en fait partie du premier groupe.

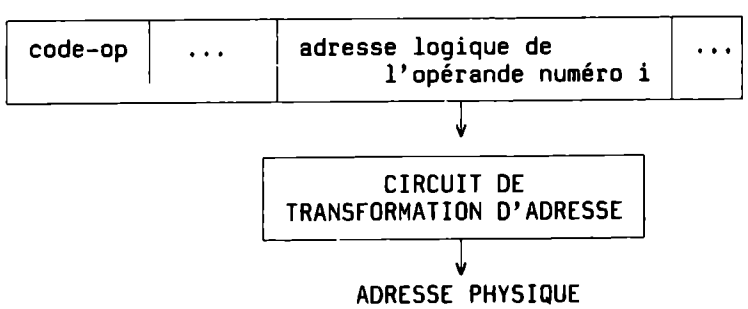


Figure 7j: place du circuit de transformation d'adresse.

7.11 – Adressage absolu.

Ce mode d'adressage est le plus évident puisqu'il consiste à avoir adresse logique et adresse effective égales: le circuit de transformation est fonctionnellement vide. Ses avantages sont les suivants:

- chaque instruction peut adresser **toute** la mémoire centrale,
- il est **rapide** puisque l'adresse logique est directement transférée à la mémoire centrale, sans être retardée par une transformation intermédiaire,
- il est **facile** à réaliser... et à comprendre par les utilisateurs puisqu'il n'introduit aucune modification de la vue qu'ils ont de la mémoire centrale.

Au titre des **défauts** nous mettrons:

- la **taille** des instructions puisque les champs adresse ont la longueur maximale. Ceci compense largement le gain que présente le circuit de transformation très simple. Cette taille extrême des instructions est d'ailleurs un facteur potentiel de ralentissement puisqu'elle conduit à un trafic plus important entre processeur et mémoire centrale;
- le manque **d'indépendance** entre les instructions et les données: considérons l'instruction LDA de code AD0205 qui charge dans l'accumulateur le contenu de la case mémoire d'adresse 0502 (toujours en hexadécimal, soit 1282 en base dix); si pour différentes raisons on ne range plus l'information concernée à cette adresse, mais à l'adresse 78A2, il faut changer l'instruction, c'est-à-dire en général faire un nouvel assemblage, une nouvelle édition des liens.

  Prenons un second exemple faisant intervenir deux tableaux dont nous voulons faire la somme:
  * le premier tableau appelé T1 est stocké à l'adresse 1700,
  * le second tableau appelé T2 se trouve lui à l'adresse 2A00,
  * le tableau résultat, somme des deux précédents est rangé à l'adresse 7700.

  Supposons que le programme est stocké à l'adresse 8000:

```
  ADRESSE   CODE   ADRESSE      COMMENTAIRE
INSTRUCTION       OPERANDE
    8000    CLC                remise à zéro de C
    8001    LDAA   0017        chargement du premier opérande
    8004    ADCA   002A        ajout du second opérande
    8007    STAA   0077        rangement du résultat
    800A    INCA   0280        préparation du traitement
    800D    INCA   0580           des entrées suivantes
    8010    INCA   0880           des trois tableaux .
    8013    JMP    0080        boucle sur CLC
```

Ce programme souffre de nombreux défauts au seul titre de la programmation, mais nous les oublierons pour l'instant. Ce qui importe c'est

de noter que l'on est forcé de **modifier les instructions** en cours d'exécution (rôle des trois INCA) afin de balayer toutes les entrées des tableaux. Non seulement il faut modifier le programme si les données sont rangées ailleurs en mémoire, mais de plus ce programme ne peut plus ré-additionner les mêmes tableaux puisque son exécution le modifie.

On notera au passage que LDA a été écrit LDAA, ADC est devenu ADCA, etc.: lorsque les instructions supportent plusieurs modes d'adressage et donc plusieurs codes opération dans les processeurs comme le 6502, on précise le mode d'adressage utilisé en ajoutant une lettre ou un symbole au nom de l'instruction. Dans le cas du 6502 nous ajouterons "A" pour l'adressage absolu.

Tous les processeurs ne disposent pas de ce mode d'adressage: les 360/370 d'IBM en sont dépourvus.

Quant à la terminologie, on trouve en américain les termes "absolute addressing", "direct addressing" pour ce mode d'adressage. Dans le 6800 de Motorola il est appelé "étendu".

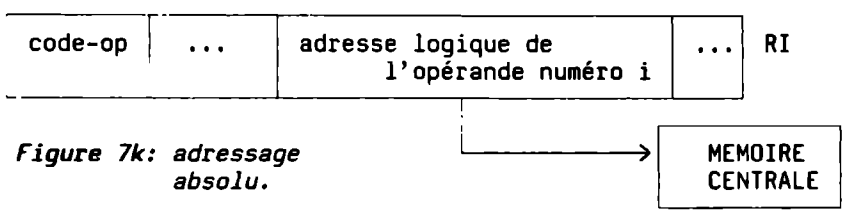


Figure 7k: adressage absolu.

7.12 – Adressage indirect.

L'adressage indirect apporte une amélioration par rapport à l'adressage absolu en créant un premier degré d'indépendance entre les instructions et les données qu'elles traitent. Le champ "adresse logique" AL de l'instruction ne contient pas l'adresse de l'opérande, mais l'adresse de son adresse et il faut au processeur deux accès à la mémoire centrale pour obtenir l'information désirée.

Le premier accès fournit l'adresse finale à laquelle est rangée la valeur de l'information. C'est donc un **pointeur** selon nos définitions et l'on observe que l'on peut très bien changer l'adresse à laquelle est rangée la valeur de l'information sans avoir à changer les instructions: il suffit de changer le pointeur. Ainsi un programme appelant un sous-programme pourra lui "passer" l'adresse de ses opérandes par un pointeur convenu à l'avance. Les pointeurs du C correspondent exactement à cette notion d'indirection.

Dans ce mode d'adressage l'adresse logique conduit à deux adresses physiques, la seconde étant l'adresse effective. Le 6502 ne possède qu'une seule instruction avec ce mode d'adressage, JMP* (l'étoile ajoutée au nom signifie "adressage indirect", tout comme la lettre "A" ajoutée à ADC signifiait "adressage absolu"). Si

```
JMP 4306
```

provoque un branchement à l'adresse 0643,

JMP* 4306

provoque un branchement à l'adresse 0769 si 6907 est le contenu des mots mémoire d'adresse 0643 et 0644 respectivement. L'instruction JMP* dans le cadre de l'exemple ci-dessous crée une boucle infinie:

```
ADRESSE MC  CONTENU  (→ : point d'entrée)
     0643   69
     0644   07
     ...
  →  0769   JMP* 4306
```

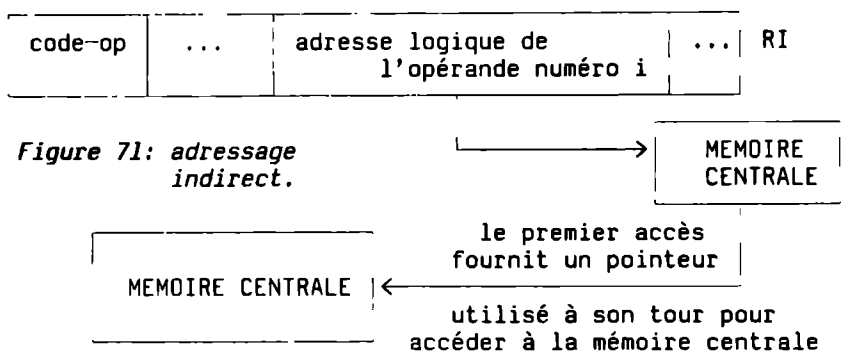


Figure 71: adressage indirect.

Nous aurons d'autres occasions d'illustrer les indirections sur le 6502. On trouve ce mode d'adressage dès 1958 sur l'IBM 709. Tous les ordinateurs ne le possèdent pas, en particulier les séries 360/370 d'IBM. De même le 8086 ne possède pas ce type d'adressage indirect. On peut cependant y utiliser des pointeurs, mais deux instructions au lieu d'une sont alors nécessaires pour obtenir l'information désirée:
- chargement d'un registre adresse,
- utilisation de ce registre pour adresser la mémoire.

Les **avantages** de ce mode d'adressage sont:
- **l'indépendance** accrue entre instructions et données,
- **l'extension** de l'espace mémoire adressable. On peut en effet imaginer que les pointeurs stockés en mémoire centrale ont plus de bits (deux fois plus par exemple) que ceux de l'adresse absolue. Dans une machine comme le 6502 on pourrait ainsi adresser 4 Go (Giga-octets). Nous verrons que des dérivés de l'adressage indirect présentent cette caractéristique dans le 6502.

Au titre des **désavantages** nous noterons:
- une **rapidité** moindre par rapport à l'adressage absolu (deux accès mémoire contre un). Mais il faut aussi tenir compte des deux instructions nécessaires pour obtenir le même résultat si l'on ne possède pas l'adressage indirect;

- une **complexité** accrue au niveau du circuit de transformation d'adresse... et de la compréhension par les utilisateurs. L'indirection se révèle toujours délicate à employer.

En ce qui concerne la terminologie, on trouve les termes de "indirect addressing", "differed addressing" dans la littérature américaine.

7.12.1 – Adressage indirect en cascade.

L'adressage indirect introduit un pointeur intermédiaire en mémoire centrale. Sa généralisation, appelée adressage indirect en cascade introduit plusieurs pointeurs intermédiaires en mémoire centrale. On le trouve sur les DPS7, DPS8, Sperry 1100, etc. Dans la plupart des cas un mécanisme est prévu pour détecter les indirections trop longues ou en boucle: limitation de la longueur de la cascade sur DPS7, sur les NOVA de DG (elles étaient initialement illimitées), limitation du temps sur DPS8.

7.13 – Adressage relatif.

Si l'adressage indirect apportait une meilleure indépendance par rapport à l'adressage absolu, l'adressage relatif apporte immédiatement une amélioration de la taille des adresses logiques. Le champ "adresse logique" AL de l'instruction contient un **déplacement** (displacement, offset en américain) qui est **relatif** à une adresse de référence (appelée aussi zéro relatif) généralement contenue dans un registre adresse que l'on appelle **registre de translation**. Le circuit de transformation d'adresse comprend donc au minimum un opérateur d'addition (additionneur, figure 7m) qui fait la somme de l'adresse logique AL et du registre de translation (RT).

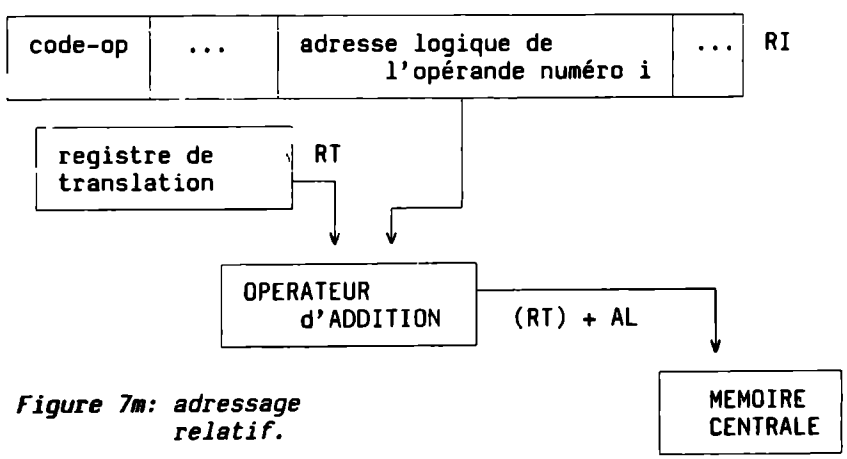


Figure 7m: adressage relatif.

Cette addition peut, selon les ordinateurs, considérer le déplacement AL comme un nombre signé (cas des 6502, 6800) ou toujours posi-

tif (cas des 360/370). La zone mémoire accessible aux instructions utilisant ce mode est donc à cheval sur le zéro relatif ou située tout entière du même côté de ce zéro relatif (figure 7n). On trouve ce mode d'adressage dès 1958 avec l'IBM 709.

L'effet de cette contraction du champ adresse logique est donc de restreindre l'espace mémoire adressable par une instruction. Les instructions du 6502 qui possèdent ce mode d'adressage sont les instructions de branchement conditionnel (BNE, BMI, BPL, etc.) et le déplacement y comporte 8 bits (instructions sur deux octets contre trois pour l'adressage absolu). Ces instructions peuvent adresser une zone comprise entre -128 et +127 par rapport au zéro relatif.

L'IBM 360/370 dispose d'un déplacement de 12 bits qui lui permet d'accéder aux adresses comprises entre zéro et 4 Ko par rapport au zéro relatif.

Lorsque l'on compare le rapport entre l'espace adressable et l'espace maximum que l'on pourrait adresser en adressage absolu, on obtient 256o/64 Ko pour le 6502 et 4 Ko/16 Mo pour les 370.

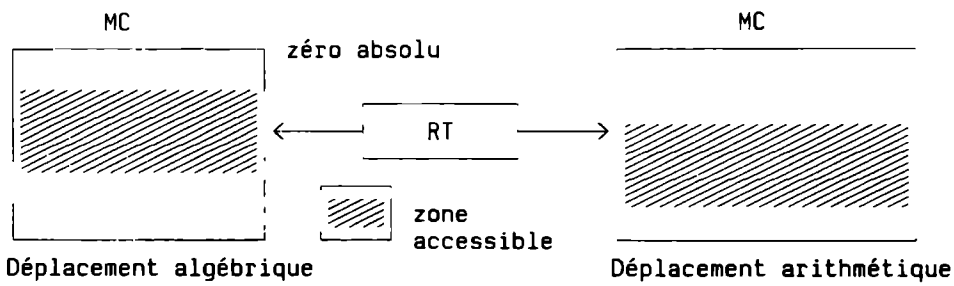


Figure 7n: zone mémoire adressable en adressage relatif.

Les **avantages** de l'adressage relatif sont:
- l'écriture d'instructions plus **courtes**, ce qui économise la mémoire centrale et crée un trafic moindre entre mémoire et processeur en ce qui concerne le flot des instructions;
- l'écriture de programmes **translatables**, c'est-à-dire de programmes qui s'exécutent de façon identique, quelle que soit l'adresse à laquelle ils sont rangés (on dira aussi chargés) en mémoire centrale;
- l'**indépendance** entre les instructions et les données puisqu'il suffit de changer le contenu d'un registre de translation pour accéder à d'autres données;
- la **protection** implicite qu'apporte cet adressage: il est physiquement impossible de sortir de la zone mémoire définie par le contenu du registre de translation et le déplacement maximum.

Pour le chapitre des **inconvénients**, nous citerons:
- l'introduction de nouvelles instructions non fonctionnelles qui ont pour rôle de charger le registre de translation RT;
- un circuit de transformation d'adresse plus complexe avec un additionneur.

7.13.1 – Adressages absolu, indirect et relatif.

Avec ces trois modes d'adressage nous avons vu les trois modes de base et il est judicieux de faire le point en présentant une analogie avec une personne qui demande où se trouve la boulangerie, ou tout autre lieu précis. Son interlocuteur lui donnera:

- une adresse **absolue** en lui disant, par exemple, "que la boulangerie est au numéro 135 de la rue de la Paix",
- une adresse **indirecte** en indiquant que le patron du café d'en face lui dira où se trouve la boulangerie. Ce patron de café joue ici le rôle de pointeur,
- une adresse **relative** en lui répondant "elle se trouve dans le second bâtiment à droite après le premier croisement". Ce croisement joue ici le rôle de zéro relatif, le second bâtiment de déplacement.

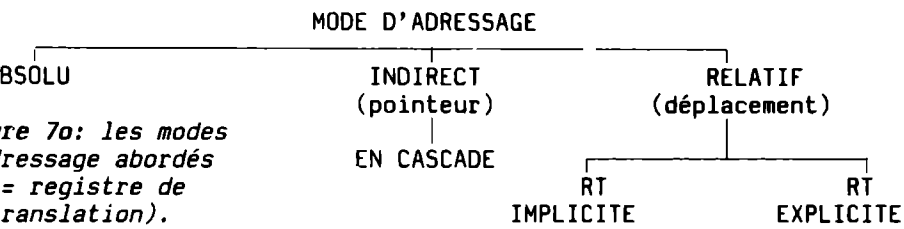


Figure 7o: les modes d'adressage abordés (RT = registre de de translation).

7.13.2 – Types de registre de translation.

Le registre de translation peut être unique (cas du 6502) ou multiple (cas des 16 registres généraux des 360/370, cas des 8 registres adresse des DPS7). Il peut être aussi:

- **implicite**: il ne figure pas dans le champ AL de l'instruction. C'est le cas du 6502 où le registre de translation est toujours le compteur ordinal CO. C'est aussi le cas des DPS7 et DPS8 pour les branchements relatifs. D'autres mécanismes que nous passerons en revue permettent de définir d'autres références implicites;
- **explicite**: il figure alors dans le champ adresse logique AL de l'instruction. Il en est toujours ainsi dans les 360/370 d'IBM, parfois dans les DPS7 de Bull.

7.13.3 – Compteur ordinal comme RT implicite.

Comme tout processeur a un compteur ordinal, il n'y a pas de registre supplémentaire à ajouter. Une autre caractéristique de l'utilisation de CO est que le zéro relatif est **mobile**: il se déplace au fur et à mesure que le programme s'exécute. Quand on se rappelle l'effet de localité déjà évoqué plus haut (les programmes ont tendance à travailler dans une petite zone, quelques-unes éventuellement, pendant un certain temps) on conçoit qu'il n'y a pas besoin de recharger le registre de translation trop fréquemment, ce qui d'ailleurs n'est pas permis pour le compteur ordinal: seuls l'enchaînement des instructions et les instructions de branchement le modifient.

7.13.4 – Adressage relatif implicite avec secteurs.

Dans ce mode d'adressage relatif, on considère que la mémoire est arbitrairement découpée en blocs de longueur fixe appelés secteurs. Dans le PDP8 de DEC les secteurs (appelés pages par DEC) avaient 128 mots de 12 bits. Le HP1000 a des secteurs de 2 ko, appelés "pages" dans la terminologie HP. La série 16 de Bull (316, 716) avait des secteurs de 512 mots de 16 bits. De même les PRIME en disposent dans un de leurs jeux d'instructions puisque ce sont au départ des séries 16.

Quand une instruction avec ce mode d'adressage se déroule, elle se trouve dans un secteur que l'on appelle courant et elle peut référencer des données qui se trouvent:
- dans le secteur courant, défini par le compteur ordinal donc,
- ou dans le secteur zéro, le premier de la mémoire.

Une instruction a donc accès à deux secteurs et lorsqu'elle ne peut atteindre une donnée dans le secteur courant, on place cette donnée (ou son pointeur) dans le secteur zéro, là où toutes les instructions ont accès (zone de mémoire commune). Ce mode apporte donc une amélioration par rapport au précédent.

On trouvait ce mode dans les mini-ordinateurs 16 bits qui avaient des mémoires de 64 Ko maximum. Dans certains le secteur zéro est devenu logique (série 16 de Bull), c'est-à-dire que tout secteur physique pouvait devenir le secteur zéro. On ne trouve plus guère ce mode d'adressage dans les processeurs récents.

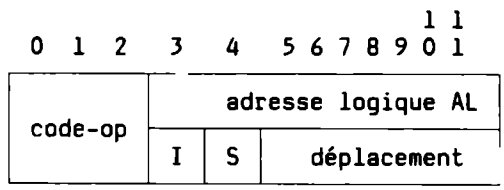


Figure 7p: format d'une instruction du PDP8.

S = secteur courant (1) ou zéro (0)
I = indirection (1) ou non (0)

Comme on peut le constater sur la figure 7p, ce mode d'adressage peut se combiner avec l'adressage indirect: la partie secteur-déplacement de l'adresse logique définit alors l'adresse d'un pointeur dans un secteur si le bit I est égal à un. L'indirection peut alors s'opérer en cascade (cas du HP1000).

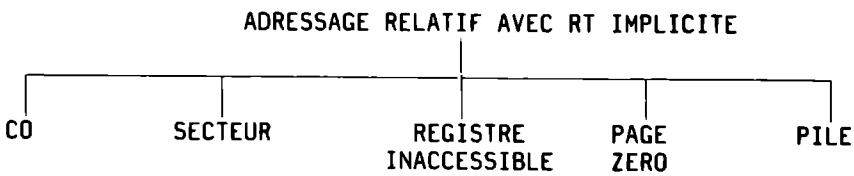


Figure 7q: types d'adressage relatif avec RT implicite.

7.13.5 – Adressage relatif implicite, RT inaccessible.

Ce mode utilise un (ou plusieurs) registre de translation inaccessible aux utilisateurs. Ce qui signifie que le processeur a au moins deux états de fonctionnement (maître/esclave évoqués précédemment) et que seul le système d'exploitation peut modifier le registre de translation RT. Ces processeurs ajoutent généralement un second registre appelé "registre longueur" RL ou équivalent (registre limite): l'ensemble des deux registres RT et RL définit une zone de mémoire en dehors de laquelle le programme ne peut pas accéder. Chaque adresse physique élaborée à partir de l'adresse logique AL est comparée à RT et RT+RL: si elle est inférieure à RT ou supérieure à RT+RL une erreur est détectée par le matériel (une interruption est générée dans la plupart des cas).

Les DPS8 de Bull, les 1100 de Sperry, les CYBER de CDC, les PDP10 de DEC ont de tels registres. Ainsi chaque programme utilisateur:
- ne peut modifier le système d'exploitation, ni le lire,
- ne peut être perturbé par un autre programme utilisateur,
- ne peut perturber un autre programme utilisateur.

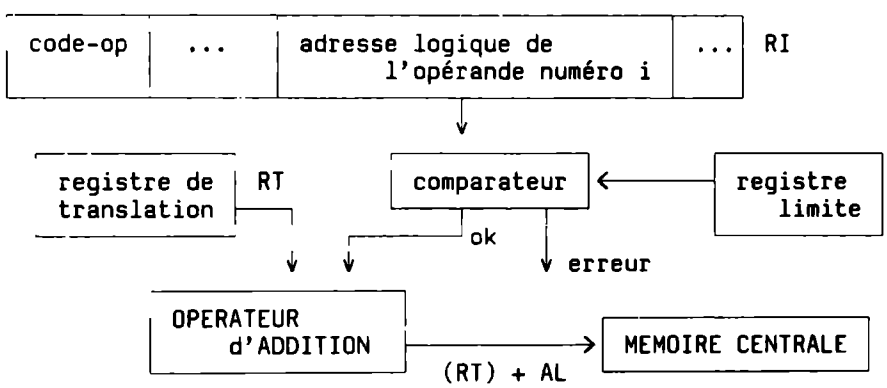


Figure 7r: protection par les registre RT et RL.

7.13.6 – Adressage relatif implicite par page zéro.

L'adressage page zéro correspond, comme l'adressage secteur, à un découpage de la mémoire centrale en un certain nombre de blocs de longueur fixe appelés "pages".

Dans le 6502 la taille des pages est de 256 octets, et la mémoire de 64 Ko peut donc contenir 256 pages. L'octet de poids fort de l'adresse est appelé numéro de page, et l'octet de poids faible le déplacement dans la page.

La page zéro est la première page de la mémoire et les instructions disposant de ce mode d'adressage seront limitées à ces 256 premiers octets. Elles ne contiennent donc dans le champ adresse logique que le seul déplacement, alors que l'adressage absolu contient le numéro de page et le déplacement (dans l'ordre inverse à l'intérieur

de l'instruction). On obtient ainsi un gain de place: dans le 6502 les instructions ayant ce mode d'adressage n'occupent que deux octets contre trois pour l'adressage absolu.

Tout comme le secteur zéro a évolué vers un secteur logique zéro dans l'adressage secteur, l'adressage page zéro a lui aussi évolué vers une page zéro logique: ainsi le 6800 de Motorola qui appelle "direct" le mode d'adressage page zéro, a eu un successeur, le 6809, dans lequel le mode appelé "absolu direct" correspond en fait à une page zéro logique. Le 6809 comporte pour cela un registre appelé "registre page" qui contient le numéro de la page utilisée comme page zéro. Ce registre est bien sûr implicite.

Nous ajouterons un "Z" au nom des instructions du 6502 lorsqu'elles utilisent ce mode d'adressage: LDAZ sera l'instruction chargeant l'accumulateur avec le contenu d'un des 256 premiers mots (des octets en fait) de la mémoire.

PROCESSEUR	PROFONDEUR DE LA PILE	EMPLACEMENT DE LA PILE	NOMBRE DE PILES
4004	3	PC	1
4040	7	PC	1
PPS4	infini	MC	1
F8	infini	MC	1
8008	7	PC	1
8080	infini	MC	1
6800	infini	MC	1
PPS8	33	PC et MC	1
68000			2
VAX			5
6502	256	MC	1

Tableau 7s: exemples de piles.

MC : mémoire centrale
PC : processeur central

7.13.7 – Adressage relatif implicite avec pile.

Les instructions qui utilisent ce mode d'adressage n'ont aucun champ adresse logique. L'adresse effective de l'opérande est implicitement le sommet de la pile.

C'est le cas des instructions PHA, PLA sur le 6502, elles n'occupent qu'un octet et sont donc très courtes. Elles utilisent implicitement un registre appelé **pointeur de pile** (S dans le 6502, SP dans le 8086) dont le contenu est modifié à chaque exécution d'une instruction avec ce mode d'adressage. Dans le 6502, la pile est obligatoirement la page 1, et S ne contient que le déplacement dans cette page.

7.13.8 – Adressage relatif avec RT explicite.

Ce type d'adressage est généralement appelé "adressage basé", et vu sa très grande utilisation, nous l'étudierons au même niveau que les adressages absolu, indirect et relatif.

7.14 – Adressage basé.

Appelé aussi "base plus déplacement", ce mode correspond à une adresse logique composée du couple [registre de base, déplacement]. Le registre de base est un registre contenant une adresse dite "de base", le zéro relatif que nous avions introduit dans l'adressage relatif. Ce qui est tout à fait logique puisque l'adressage basé est un cas particulier de l'adressage relatif.

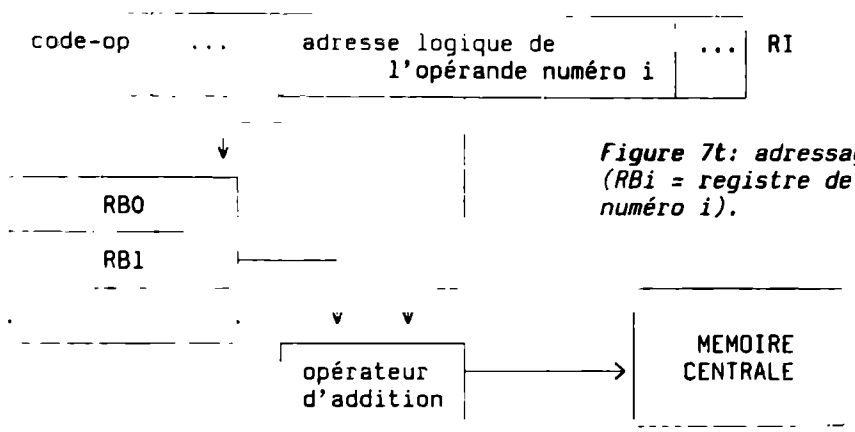


Figure 7t: adressage basé (RBi = registre de base numéro i).

La figure 7x donne l'exemple d'une instruction avec deux opérandes dont l'adresse logique est de type basé. On retrouve les quatre bits pour définir l'un des 16 registres généraux qui peuvent aussi jouer le rôle de registre de base, ainsi que les douze bits du déplacement qui permettent de travailler sur une zone mémoire de 4 Ko. Si l'on désire changer la zone rendue accessible par ce registre, il faut changer le contenu du registre de base. Le 370 n'ayant ni adressage absolu, ni adressage indirect, il faut au moins deux instructions là où les deux autres modes se contentent d'une seule instruction:
- une instruction (ou deux) de chargement d'un (ou deux) registres de base,
- l'instruction de manipulation des données.

```
LA  2,OP1          range l'adresse de OP1 dans RB2
LA  3,OP2          range l'adresse de OP2 dans RB3
MVC 0(27,3),0(2)   copie 27 caractères de OP1 dans OP2
```

Figure 7v: exemple de copie de chaînes sur un 370.

Heureusement, un chargement de registre de base permet généralement a de nombreuses instructions de manipulation de pouvoir s'exécuter. La figure 7w le montre par rapport à la figure 7v, dans le cas où

les deux chaînes sont dans le même bloc de 4 Ko.

```
LA  2,ADRDONNEES       charge dans RB2 l'adresse des données
MVC OP2(27,2),OP1(2)   copie les 27 caractères de OP1 dans OP2
```

Figure 7w: utilisation multiple d'un registre de base (OP1 et OP2 sont des déplacements dans ce cas).

L'instruction de la figure 7x est dite du type SS (Storage to Storage) car ses deux opérandes sont en mémoire centrale. C'est le cas des instructions de déplacement de chaînes de caractères comme MVC, le champ "longueur" indiquant le nombre d'octets déplacés. Avec les deux fois 32 bits des deux instructions de chargement des registres de base RB1 et RB2, un tel déplacement de caractères demande 112 bits d'instruction.

```
                 111111   1111   222222222233   3333   333344444444
01234567 89012345   6789   012345678901   2345   678901234567
```

code-op	longueur	opérande numéro 1		opérande numéro 2	
		B1	D1	B2	D2

Bi : registre de base de l'opérande numéro i
Di : déplacement de l'AL de l'opérande numéro i

Figure 7x: format d'une instruction de type SS dans les 370.

L'adressage basé est donc une généralisation de l'adressage relatif avec registre implicite. Il multiplie le nombre de zones accessibles (16 fois 4 Ko, soit 64 Ko sur un 370, autant que l'adressage absolu sur un 6502). La multiplicité des registres de base correspond aussi à une évolution des logiciels déjà sensible à l'époque de la conception du 360 (fin des années 50, début des années 60): les programmes sont de moins en moins des ensembles compacts de code et données mélangés, mais des ensembles de modules avec des caractéristiques particulières, rangés de façon indépendante en mémoire. L'utilisation des pointeurs associés aux structures illustre en C cette notion d'adressage basé.

7.15 – Adressage indexé.

L'adressage indexé présente de nombreuses analogies avec l'adressage basé, et la figure 7t reste valable en changeant les registres de base RBi en registres d'index RXi (ce sont d'ailleurs les mêmes sur 370). La grande différence entre les deux modes réside dans l'utilisation qui est faite des registres:

- le registre de base contient une **adresse**,
- le registre d'index contient un **déplacement**, utilisé pour indicer un tableau, une structure plus généralement.

Le registre de base voit son contenu évoluer beaucoup plus rarement que celui du registre d'index. A tel point que dans la 1100 de Sperry un des bits de l'adresse logique indique s'il faut ou non incrémenter le registre d'index après exécution de l'instruction: le programme est plus court puisqu'il n'y a pas nécessairement d'instruction de chargement et incrémentation du registre d'index, il est aussi plus rapide puisque l'incrémentation du registre d'index se fait en parallèle avec le reste de l'exécution de l'instruction. La 1100 visait initialement le marché scientifique qui est très gourmand de tableaux.

Ce mécanisme "d'auto-modification" des registres d'index se retrouve aussi sur les PDP11, VAX, 68000 de Motorola, sous le nom d'adressage indexé avec "auto-incrémentation". Il est d'ailleurs généralisé de deux façons:

- on peut soit incrémenter, soit décrémenter le registre;
- l'opération peut se faire avant ou après la transformation d'adresse, réalisant ainsi des pré-incrémentation (tab[++i] en C), post-incrémentation, pré-décrémentation, post-décrémentation (tab[i--] en C).

Ainsi, sur le 68000, l'instruction

MOVE (A1)+,(A2)+

permettra, avec les instructions procédurales nécessaires, de copier les différentes entrées d'un tableau, les registres A1 et A2 étant incrémentés après transfert. Le 6502 a deux registres d'index X et Y de huit bits. Dans le programme ci-dessous:

CODE-OP	2-EME OCTET	3-EME OCTET	COMMENTAIRE
A2	0C		charge 12 (0C en hexa) dans X
FE	33	22	incrémente le mot mémoire d'adresse 2233 + 0C soit 223F

on constate que l'adressage indexé utilisé relève plus exactement de la catégorie "absolu indexé" puisque le contenu du registre d'index est ajouté à l'adresse absolue 2233. Nous ajouterons un "X" au nom des instructions lorsqu'elles utilisent l'adressage indexé. Ainsi le code opération FE correspondra à l'instruction INCAX avec INC comme "incrémentation" et A comme "adressage absolu". L'adressage indexé peut aussi s'ajouter à l'adressage page zéro et les instructions utilisant cette combinaison auront un nom terminé par "ZX" ou "ZY" selon le registre utilisé: LDYZX chargera le registre d'index Y, STAZX rangera en mémoire le contenu de l'accumulateur:

CODE-OP	2-EME OCTET	COMMENTAIRE
A2	17	met 17 dans X
B4 (LDYZX)	E0	charge dans Y le mot mémoire d'adresse F7
A9	15	met 15 dans A (l'accumulateur)
95 (STAZX)	F0	range le contenu de A à l'adresse 07 car l'addition F0 + 17 est faite modulo 256 !

Les DPS8 de Bull disposent de 8 registres d'index de 18 bits qui s'ajoutent au déplacement de 18 bits. Les séries 360/370 ne disposent pas du seul adressage indexé, il est toujours combiné à l'adressage basé vu plus haut. Il en est de même dans les DPS7.

Si nous reprenons l'exemple de la copie des deux chaînes qui prenait 112 bits d'instruction dans un 370, nous aurons sur 6502 l'exemple ci-dessous qui, lui, nécessite 88 bits:

```
       LDX#    1A        range 26 dans le registre d'index X
SUITE: LDAAX   OP1       charge dans A l'octet d'adresse effective
                         OP1 plus X
       STAAX   OP2       range l'octet de A à l'adresse effective
                         OP2 plus X
       DEX               décrémente X
       BPL     SUITE     va à l'adresse SUITE tant que X est positif
```

7.16 – Adressage base plus index.

Ce mode regroupe l'ensemble des deux modes vus précédemment: l'indexation vient s'ajouter aux effets de l'adressage basé. Le circuit de transformation opère une double addition:

adresse effective = (RBi) + déplacement + (RXj)

comme le montre la figure 7y, où le format de l'adresse logique AL est donné figure 7z.

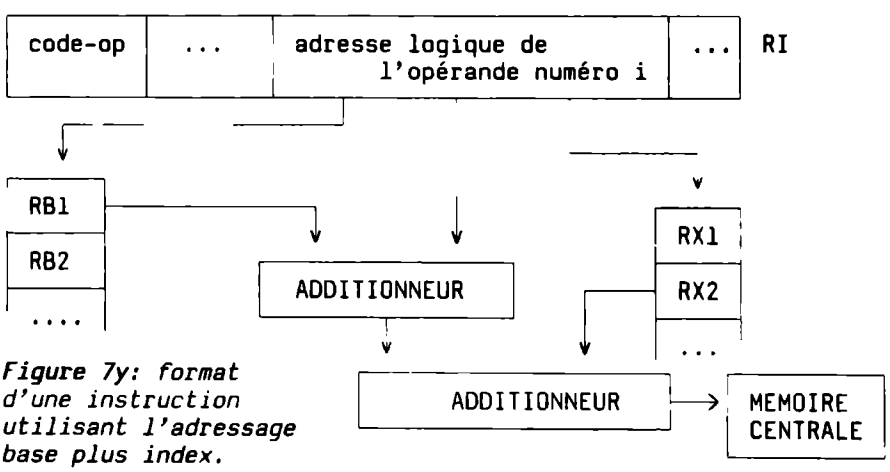


Figure 7y: format d'une instruction utilisant l'adressage base plus index.

De nombreuses machines utilisent cette technique: les 360/370 d'IBM, les DPS7 de Bull où registres de base et d'index sont physiquement différents, les DPS8 si l'on prend en compte le registre de base d'accès protégé ou les instructions EIS. Nous assimilerons à ce mode l'adressage dit "double index" que l'on rencontre sur des processeurs comme le 8088 où l'on peut ajouter le contenu des registres BX ou BP au contenu de SI ou DI pour former une adresse.

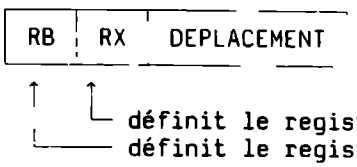


Figure 7z: format d'une adresse logique AL base plus index.

7.17 – Adressage base plus index plus indirection.

Les modes d'adressage se combinent souvent: nous venons de voir l'adressage base plus index, nous avions déjà vu que l'adressage indexé se combinait aussi facilement avec l'adressage absolu, l'adressage page zéro, etc. Il peut aussi se combiner avec l'adressage indirect et comme nous allons le voir ici, avec l'adressage base plus index.

La combinaison de ces trois modes se présente sous deux variantes selon que:

- l'indexation se fait avant l'indirection (figure 7aa). C'est le cas des 6502, des DPS8. On parle alors de **pré-indexation;**
- l'indexation se fait après l'indirection (figure 7ab). C'est le cas général lorsque les deux variantes ne sont pas disponibles et on la trouve sur les DPS8, DPS7, 6502, ... mais pas sur les 370 qui n'ont pas d'indirection du tout. On parle alors de **post-indexation.**

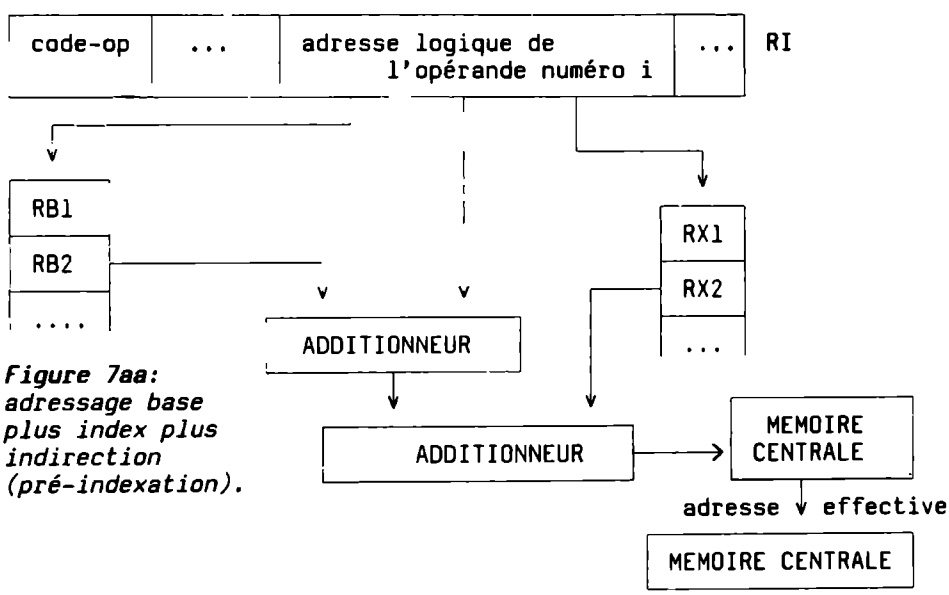


***Figure 7aa:** adressage base plus index plus indirection (pré-indexation).*

Dans les exemples utilisant le 6502 nous prendrons le symbole "*" pour représenter l'indirection, comme nous l'avons déjà illustré dans le cas de JMP. LDA*Y chargera l'accumulateur avec un adressage post-indexé par Y, alors que LDAX* le chargera avec un adressage pré-indexé

par X. Le 6502 n'offre pas réellement un adressage base plus index plus indirection dans la mesure où il ne possède pas de registre de base, mais il permet cependant de bien illuster les mécanismes de base de la pré et post-indexation.

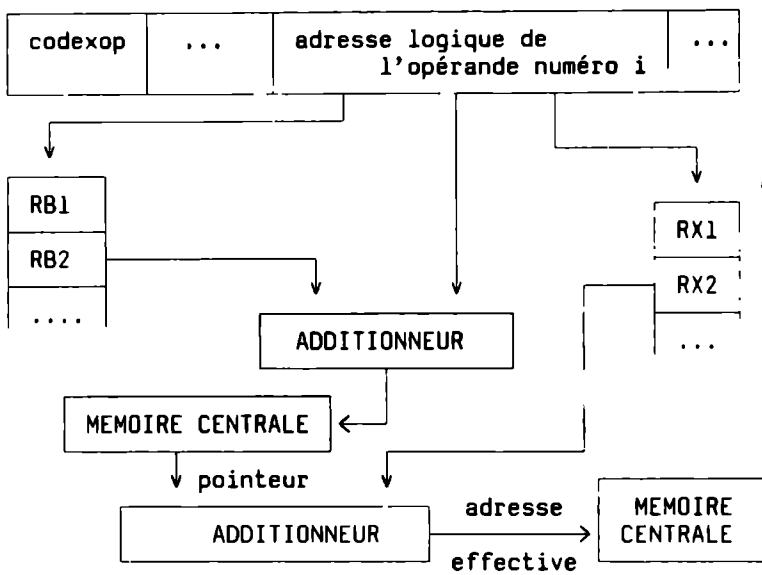


Figure 7ab: post-indexation.

Le programme ci-dessous, dont les instructions se trouvent aux adresses 2728 et suivantes:

```
ADRESSE MC    CONTENU    COMMENTAIRE   (→ : point d'entrée)
   0029         81       déplacement dans la page
   002A         07       numéro de page
   ....
   0781         4E       valeur chargée dans l'accumulateur
   ....
 → 2728         A2       met dans X
   2729         08            la valeur décimale 8
   272A         A1       LDAX*
   272B         21            opérande de LDAX*
```

mettra 4E dans l'accumulateur A, de la façon suivante:
- le déplacement 21 à l'adresse 272B définit une adresse dans la page zéro,
- l'indexation ajoute le contenu de X, soit 8, à cette adresse, d'où 29 qui correspond à l'adresse d'un pointeur,
- ce pointeur est égal à 0781 comme on peut le voir dans les cases mémoires d'adresse 2A et 29,
- l'instruction se termine en mettant le contenu de la case d'adresse 0781 dans l'accumulateur.

7.18 – Adressage immédiat.

Avec l'adressage immédiat nous abordons un mode totalement différent des précédents puisque le champ adresse logique AL contient la valeur de l'information. Or tous les modes vus ci-dessus contenaient une adresse dans AL. Ce sont généralement des **constantes** qui sont ainsi utilisées, c'est-à-dire des informations dont la valeur ne change pas durant l'exécution du programme. Il est en effet de moins en moins fréquent de modifier les instructions en cours d'exécution, nous aurons l'occasion de développer ce point dans le Tome V lorsque nous aborderons les sous-programmes invariants, ré-entrants, etc.

Le 6502 dispose de ce mode et les instructions qui l'utilisent ont un dièse # ajouté à leur nom:

LDY# 1A

mettra 26 en décimal dans le registre Y.

Le circuit de transformation d'adresse n'est pas sollicité dans ce mode d'adressage.

Selon les ordinateurs, on parlera d'adressage immédiat **court** ou **long** lorsque le champ AL peut prendre différentes tailles. Le 8086 offre 8 et 16 bits, le 68000, le Z8000 et le NS16032 offrent 8, 16 et 32 bits. Sur DPS7 on trouve aussi des champs de 4 bits. Peu d'instructions du 370 disposent de ce mode, les MVI, CLI permettent de copier, comparer un octet; l'instruction de chargement d'adresse LA permet aussi de manipuler indirectement des constantes.

7.19 – Adressage registre.

On trouve fréquemment, depuis l'arrivée des microprocesseurs en particulier, le terme d'adressage registre:

- adressage registre **direct**: le champ AL contient le numéro du registre dans lequel se trouve la valeur de l'information manipulée. C'est le cas de l'instruction AR R1,R2 sur 370, instruction de type RR (Registre à Registre). C'est aussi le cas des MOV codées 1000100w 11dddsss sur 8086 où ddd et sss sont deux numéros de registre;
- adressage registre **indirect**: le registre dont le numéro se trouve dans AL contient l'adresse de l'information, et la valeur est bien en mémoire centrale. On peut encore dans ce cas (celui du 8086 par exemple) distinguer deux sous-modes selon qu'il y a un champ déplacement ou non dans AL; on retrouve en fait ce que nous avons appelé adressage **basé** ci-dessus. C'est dans cette catégorie qu'INTEL range aussi le mode base plus déplacement.

7.20 – Autres termes.

- adressage **préfixé**: terme utilisé dans les multiprocesseurs 370 d'IBM où chaque processeur doit disposer de 4 Ko qu'il est seul à pouvoir accéder;
- adressage **implicite**: le mode d'adressage est imposé par le code opération;
- adressage relatif par rapport aux **vecteurs d'interruption** dans les DPS6;

- adressage externe sur NS16000 lié au problème de références entre modules de programmes;
- adressage linéaire lorsque l'espace adressable est vu comme un ensemble continu avec des cases numérotées de 0 à N, etc.

7.21 – Adressage mot et caractère.

Nous avons dit que le processeur envoyait à la mémoire centrale des adresses de "cases". Il y a en fait deux grands types de case, les **caractères** et les mots.

Les machines récentes adressent des caractères: IBM 360, DPS7, VAX, etc. Un mot de 32 bits est alors défini comme 4 octets commençant à l'adresse caractère définie. Ceci ne signifie pas que les mémoires centrales de ces machines sont physiquement organisées en caractères; elles sont toujours organisées en mots ou même en double mots sur le plan physique, mais les instructions peuvent faire commencer les mots à n'importe quelle adresse de caractère. Si cela est bien vrai pour les IBM 370, ce ne l'était pas pour les 360.

Dans une machine où les champs AL définissent des adresses de mot, la manipulation des caractères est plus délicate puisque l'on ne sait obtenir ou ranger que des mots en mémoire centrale. On trouvera donc dans les processeurs 16 bits des instructions qui inversent les deux octets (SWAB sur PDP11), en mettent un à zéro afin de simplifier les comparaisons. Le DPS8 de Bull relève de ce type dans son jeu d'instructions initial.

A la fin de ce chapitre 7 nous avons une vue du sous-système central résumée figure 7ac: le processeur central comprend trois parties:
1. une consacrée aux données,
2. une autre dédiée aux adresses,
3. la dernière spécialisée dans l'enchaînement des instructions.

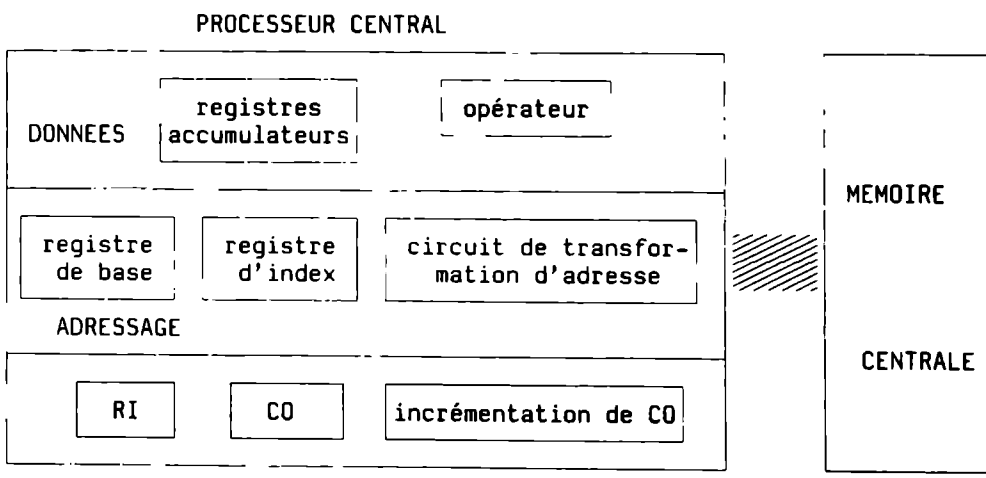


Figure 7ac: structure actualisée du sous-système central.

chapitre 8

typologie des ordinateurs

8.1 – Critères de classement.

Il existe de nombreuses manières de classer les ordinateurs, beaucoup étant basées sur les caractéristiques du sous-système central. On utilisera ainsi:

- la vitesse, la puissance,
- le type (micro, mini, etc),
- le coût,
- le domaine d'utilisation (scientifique, gestion, temps réel, CAO, génie logiciel, etc),
- le nombre de bits du mot (machine 32 bits),
- le volume. Un classement établi en 1975 par M. Lilen conduisait au résultat suivant en se basant sur le volume de l'unité centrale:
  - * 10 m^3 pour les superordinateurs,
  - * 1 m^3 pour les "maxi-ordinateurs" (ordinateurs classiques),
  - * 0,5 dm^3 pour les "midi-ordinateurs",
  - * 100 dm^3 pour les minis,
  - * 80 dm^3 pour les "milli-ordinateurs",
  - * 50 dm^3 pour les micros.
- etc.

8.2 – Micros, minis, moyens, etc.

Les classements basés sur les termes de micro, mini, etc, font intervenir des critères qui ne sont pas toujours directement comparables. Mais ils sont malgré tout très utilisés et expriment une idée générale de puissance.

Le terme de **micro-ordinateur** correspond à un ordinateur dont le processeur central est implémenté sur une **puce**. C'est donc un critère purement technologique qui traduit cependant une idée de faible puissance de traitement. Idée de moins en moins vérifiée avec le temps car les microprocesseurs de haut de gamme sont très rapides, même par rapport à des minis et des moyens ordinateurs.

Le terme de **mini-ordinateur** est plus ancien. Un mini-ordinateur était autrefois une machine tout simplement plus petite qu'un gros ordinateur ou ordinateur tout court. Il avait des mots de 12 bits

(PDP8) ou 16 bits (série 16 de Bull, PDP11, ECLIPSE de Data General, MITRA 15 de l'ex-CII, SOLAR 16 de SEMS, etc.), une mémoire centrale ne dépassant jamais 64 Ko. Il était très utilisé dans le contrôle de processus (systèmes temps réel) et dans l'éducation avec des systèmes d'exploitation interactifs alors que les gros ou moyens ordinateurs avaient des systèmes d'exploitation orientés vers le traitement par lot (nous expliquerons tous ces termes dans le Tome II). Ils étaient encore considérés comme parents pauvres en logiciel dans les années 75 à 80. Leur architecture interne était orientée autour d'un bus, ce qui n'était pas le cas des autres. De plus les minis ne requéraient généralement pas l'environnement exigeant des autres ordinateurs: salle avec température, air, humidité régulés.

Aujourd'hui la situation est moins nette. Les minis sont talonnés par les micros (certains minis ont d'ailleurs une version micro (LSI11 pour les PDP11, les micro-VAX, etc.) et ils talonnent les ordinateurs moyens (43XX d'IBM, DPS7 de Bull). La taille de leurs mots est égale à celle des autres (32, voire 48 bits sur Harris 800), et même les gros ordinateurs vont jusqu'à le clamer lorsque leur architecture est basée sur un bus ! De plus tous les systèmes d'exploitation évoluent vers l'interactif dans la mouvance de MULTICS. De synonyme de pauvreté, les minis sont passés au stade de référence: IBM parlera de supermini pour son haut de gamme 43X1, CDC traitera de même son bas de gamme de la série 800.

En fait lorsque l'on parle aujourd'hui de minis, on parle de produits commercialisés par des constructeurs ayant l'image de fabricants de minis (DEC, PRIME, Data General, Norsk Data, etc.).

Nous n'insisterons pas sur la définition des mégamini, supermini qui se veulent des minis très puissants. Ces problèmes de définition sont fréquemment évoqués et une étude récente de Computerworld (30 sept. 85) consacrée aux minis évoque ces problèmes, page SR/2 pour les minis ("What is a minicomputer") et SR/4 pour les superminis qui occuperaient l'espace entre les minis et les "mainframe".

L'ordinateur dit **moyen** est généralement un ordinateur de gestion, constituant le bas et le milieu d'une gamme comme la gamme 370. Il faut noter d'ailleurs que cette gamme 370 s'est partagée en deux: les ordinateurs moyens sont devenus les 43XX, les gros ont donné les 308X puis les 3090-X00.

Les gros ordinateurs sont des hauts de gamme, aussi bien en gestion qu'en calcul scientifique, encore que dans ce second domaine les différenciations soient plus nettes. Lorsque l'on utilise le terme de **processeur vectoriel** on vise deux catégories de machine:

- des processeurs périphériques, sortes de gros opérateurs, que l'on raccorde à un ordinateur sous forme de sous-système périphérique,
- des ordinateurs avec de très grandes capacités de calcul (CRAY 1, CYBER 205, DENELCOR, etc.)

Les Américains parlent dans ce cas de "number crunchers".

Ils utilisent aussi le terme de **mainframe** qui tend à reconstruire l'ancienne distinction entre minis et gros/moyens ordinateurs. Certains associent à ce terme des contraintes liées au refroidissement: à la différence des autres, les "mainframes" ont des dispositifs de refroidissement très élaborés.

Quant aux termes de serveur ou hôte (host), ils définissent une fonction plus qu'une idée de taille, de puissance. On parle d'ailleurs de micro-serveurs.

Les puissances de ces différentes catégories d'ordinateurs glissent vers le haut avec le temps. [INM83] place les "mainframe" entre 0,4 et 1 MIPS au début des années 60, entre 7 à et 12 MIPS au début des années 70, ce qui semble excessif, entre 8 et 25 en 1983; quant aux minis il leur attribue 0,5 à 1,5 MIPS au milieu des années 70 et 0,7 à 1,5 au début des années 80.

8.3 – Machines à n bits.

L'arrivée des minis à mots de 32 bits, puis celle des microprocesseurs à mots de 4, 8, 16 et 32 bits aujourd'hui a créé cette expression de "machines à n bits". Une machine à n bits dispose de:
- mots de n bits,
- registres de n bits,
- opérateurs travaillant sur n bits simultanément,
- un chemin de données de n bits entre le processeur et la mémoire centrale.

PROCESSEUR	TAILLE MOT	TAILLE REGISTRE	TAILLE UAL	LARGEUR DU CHEMIN DES DONNEES	MULTIPLEXAGE DONNEES ET ADRESSES
machine n bits	n	n	n	n	non
CRAY 1	64	64			
CYBER 170	60				
DPS8	36	36	72	72	non
VAX-11/780	32	32	32	32	oui
VAX-11/730	32	32	32	32	oui
68000	32	32	16	16	
68020	32	32	32		
8086	16	16	16	16	
8088	16	16	16	8	
8080	8	8	8	8	
6502	8	8	8	8	non

Table 8a: taille des éléments de différents ordinateurs.

L'idée sous-jacente est une idée de puissance: une machine n bits traite n bits d'information en parallèle, c'est-à-dire plus d'information qu'une machine p bits, p étant inférieur à n. Comme le montre le tableau 8a, certains processeurs se trouvent dans une situation intermédiaire. C'est ainsi que le 68000 sera parfois appelé un 16/32, 16 en externe, 32 en interne.

Nous proposons ci-après un exemple permettant de comparer un microprocesseur 8 bits (le 6502) à un 16/32 (le 68000), en prenant une séquence d'instructions copiant 1024 (1 Ko) octets.

```
Cas du 6502 :
       LDA#  00        ranger deux pointeurs aux adresses
       STAZ  00        0 et 2 en MC : pour copier 1024 octets
       STAZ  02        de l'adresse 8000 à l'adresse A000
       LDA#  80
       STAZ  01
       LDA#  A0
       STAZ  03
       LDX#   4        pour copier quatre fois 256 octets
       LDY#   0        pour copier les 256 octets d'une page
SUITE: LDA*Y 00        charge dans A l'octet à copier
       STA*Y 02        copie l'octet
       INY             on passe à l'octet suivant
       BNE   SUITE     Y évolue modulo 256; tant qu'il ne repasse
                       pas par zéro, on boucle dans la même page
       INCZ  01        incrémente le pointeur source (la page)
       INCZ  03        incrémente le pointeur destination (la page)
       DEX             a-t-on copié la dernière page?
       BNE   SUITE     non
       .... .....      oui
```

Cette séquence occupe 32 octets et nécessite 16 466 cycles, soit 16,5 ms avec un 6502 dont l'horloge est à 1 MHz.

Si nous considérons maintenant le 68000, la copie sera réalisée par une instruction MOVE. On utilisera MOVE.B, MOVE.W ou MOVE.L selon que l'on copie par octet, par mot de 16 bits ou par mot de 32 bits (mot long). Considérons ce dernier cas qui est le plus favorable au 68000 et est environ trois fois plus rapide que le cas MOVE.B:

```
       LEA.L   #$8000,A0     charge dans A0 l'adresse source
       LEA.L   #$A000,A1     charge dans A1 l'adresse destination
       MOVE.W  #$0100,D0     charge dans D0 le nombre de mots
                             longs à copier (1000/4=256)
SUITE: MOVE.L  (A0)+,(A1)+   copie un mot et post auto-incrémente
       DBRA    D0,SUITE      décrémente D0 et boucle s'il est positif
```

Cette séquence occupe 18 octets, presque deux fois moins par rapport au 6502. Avec une horloge à 8 MHz (huit fois plus que le 6502), les 7704 (24+30*256) cycles nécessaires prennent 0,96 millisecondes, soit seize fois moins environ que le 6502. Quand on considère le rapport des cycles (8), on voit que le 68000 est à technologie égale deux fois plus rapide que le 6502 sur cet exemple.

8.4 – Gestion, scientifique, temps réel, universalité.

Cette classification fait apparaître le **domaine** d'utilisation de l'ordinateur. Un mini de gestion sera par exemple un mini utilisé dans des applications de gestion (facturation, gestion de stock, gestion du personnel, etc.). Il disposera de langages comme le Cobol ou le Gap (RPG) ou d'environnements moins ouverts mais plus efficaces dans leurs domaines d'utilisation. Les HP 3000, les DPS1 et DPS4 de Bull, les IBM 32, 34, 36, 38 ont une très nette image de machine de gestion.

Lorsqu'il s'inscrira dans un ensemble plus complexe (réseaux d'ordinateurs) où il coopère avec de plus gros ordinateurs, on parlera de minis orientés vers le **traitement distribué** (créneau visé par le 8100 d'IBM par exemple), d'informatique "départementale" (S/36 d'IBM).

Parallèlement au mini de gestion on trouvera des minis plus adaptés au temps réel, c'est-à-dire au pilotage de processus industriels. Dans cette catégorie nous citerons les PDP11, les NORSKDATA, les HP 1000, l'IBM Série/I, etc.

La tendance à l'universalité étant très répandue, de nombreux constructeurs qualifieront leurs minis **d'universels** (general purpose) comme DEC pour son VAX: ils sont supposés bien traiter tous les types de problèmes, mais en fait ils manquent souvent des environnements transactionnels (cf. Tome II) sophistiqués et à haute sécurité de fonctionnement que possèdent les moyens et gros ordinateurs de gestion.

Si nous avons débuté cette section par des exemples concernant les minis c'est pour insister sur le fait que les premiers ordinateurs étaient ainsi très différenciés: il y avait des ordinateurs de gestion et des ordinateurs scientifiques et certains constructeurs comme IBM maintenaient des séries (ce n'étaient pas encore des gammes) d'ordinateurs séparées pour ces deux domaines. Avec la série 360 on a vu apparaître les ordinateurs **universels**, c'est-à-dire traitant aussi bien les problèmes de gestion que les calculs scientifiques. Des constructeurs comme CDC ont raté ce virage et ont occupé le haut de gamme du calcul scientifique, haut de gamme où IBM n'a pu se placer jusqu'ici.

Avec la miniaturisation des composants de tout type (puces, disques, écrans, etc) les ordinateurs envahissent de plus en plus de domaines d'activité. C'est le cas des machines **bureautiques** utilisées pour le traitement de texte, le classement, la messagerie, etc.

Dans le domaine des réseaux, la numérisation et la commutation de paquets ont fait apparaître des ordinateurs dans de nombreux réseaux, publics ou privés, télé-informatiques ou téléphoniques.

8.5 – Micro professionnel, domestique.

Les micro-ordinateurs se sont très vite diversifiés en deux grandes catégories:

- les micros domestiques (OI comme ordinateur individuel) sont généralement des ordinateurs de jeux, valant quelques milliers de francs en 1985. L'écran de télévision peut leur servir d'équipement de visualisation. Le marché français a d'abord été dominé par des produits principalement anglais. Avec la norme MSX (constructeurs japonais et Micro-Soft) l'état du marché va sans doute évoluer;
- les micros professionnels, type PC (comme personal ou professional computer), valent généralement plusieurs dizaines de milliers de francs et sont utilisés dans les bureaux, dans le milieu du travail. Des modèles portables avec écran et deux disquettes sont cependant accessibles à des prix de douze mille francs à mi-85.

Etant donné la petite taille de ces micros qui décroît sans cesse, on distingue encore des sous-catégories aux frontières intuitives: portables, portatifs.

8.6 – Stations, postes de travail.

L'évolution des ordinateurs tend à répartir de plus en plus l'intelligence, c'est-à-dire en fait à donner de plus en plus d'autonomie, de pouvoir, de décision, aux différentes parties du système informatique. Cette tendance se retrouve aussi au niveau des postes de travail.

Le poste, la station de travail, est initialement un équipement "bête" (dumb terminal) qui permet d'accéder à un sous-système central où tout se fait, tout se décide. Aujourd'hui, en particulier dans le domaine scientifique, le poste de travail de haut de gamme se situe entre les supermicros et les minis ordinateurs; il correspond à des prix de 160 à 600 kF et est commercialisé par des sociétés comme Apollo, DEC (microVAX II), HP (HP9000 sous UNIX), Sun. Longtemps absent de ce secteur, IBM a récemment annoncé (janvier 1986) le 6150 sous UNIX.

8.7 – Ordinateurs monocartes (SBC).

Pour les "intégrateurs" de système (chapitre 14) il existe des sous-systèmes centraux complets sur une seule carte (Single Board Computer), carte qui englobe la logique de commande des périphériques. La revue américaine Mini-Micro dressait en juin 85 une liste de plus de 150 produits, 46 basés sur le 68000 de Motorola, 34 basés sur le Z80.

8.8 – Les prix.

La gamme des prix est très étendue et va d'un peu plus de 100 $ pour les SBC à plusieurs dizaines de millions de dollars pour les gros ordinateurs.

Le tableau 8c extrait du Monde Informatique du 13 janvier 86 donne la structure du parc français en fonction du prix des systèmes informatiques.

	PRIX	NOMBRE AU 1.1.85	NOMBRE LIVRES EN 85	CROISSANCE ANNUELLE
ORDINATEURS DOMESTIQUES		825 000	410 000	
MICRO-ORDINATEURS	10 à 130kF	362 000	250 000	37 %
PETITS SYSTEMES	0,13 à 1,6MF	97 000	28 000	24,5 %
MOYENS SYSTEMES	1,6 à 7 MF	3 600	1 050	4,5 %
GRANDS SYSTEMES	> 7 MF	1 750	465	8,4 %

Tableau 8c: structure du parc français des ordinateurs au 1.1.85.

chapitre 9

organisation interne du sous-système central

9.1 – Bus, registres, opérateurs, chemin des données.

Nous abordons dans ce chapitre l'architecture interne du sous-système central que nous n'avions fait qu'effleurer dans les chapitres précédents. Les informations, au sens large (données, opérandes, instructions, adresses), manipulées par le sous-système central sont:
- transférées sur des bus,
- mémorisées dans des mémoires, des registres,
- traitées par des opérateurs.

L'ensemble de tous ces circuits et des liaisons qui les relient forment ce que l'on appelle le **chemin des données** (data path en américain).

Si l'architecture externe fait intervenir la mémoire centrale, les registres adressables, les opérateurs par le biais des codes opération, elle n'a pas à se préoccuper de la façon dont ces objets:
- sont organisés, interconnectés,
- dialoguent entre eux.

Ces choix ne remettent pas en cause le fonctionnement externe de l'ordinateur, ils sont dictés par les contraintes de coût, de performance que l'on veut obtenir, en tenant compte de la technologie du moment. Ainsi l'architecture interne d'un 4321 est plus simple, moins coûteuse que celle d'un 4361, et le 4321 est moins puissant.

La puissance n'est pas obligatoirement liée à la simplicité et au coût de l'architecture interne. Si un constructeur comme IBM peut se permettre de réaliser une architecture interne très différenciée pour chaque modèle de sa gamme, grâce au grand nombre de modèles vendus, les autres constructeurs n'ont pas toujours des modèles moins puissants, simultanément plus simples et moins coûteux que leur haut de gamme. Des raisons commerciales interviennent: il faut offrir une gamme étendue et certains modèles moins puissants sont en fait des modèles haut de gamme bridés, ce qui peut les rendre plus complexes au niveau de l'architecture interne. On a même vu des machines bi-vitesse comme les 66/X7 (prédécesseurs des DPS8) conçues pour fonctionner plus rapidement en temps partagé qu'en traitement par lots.

9.2 – Organisation des registres.

Il y a trois grands types d'organisation pour les registres:

- **registres indépendants**: chaque registre dispose de ses propres circuits de commande, c'est-à-dire des circuits qui permettent d'y ranger ou d'y lire l'information. Nous représenterons ces circuits de commande sous la forme donnée en figure 9a.

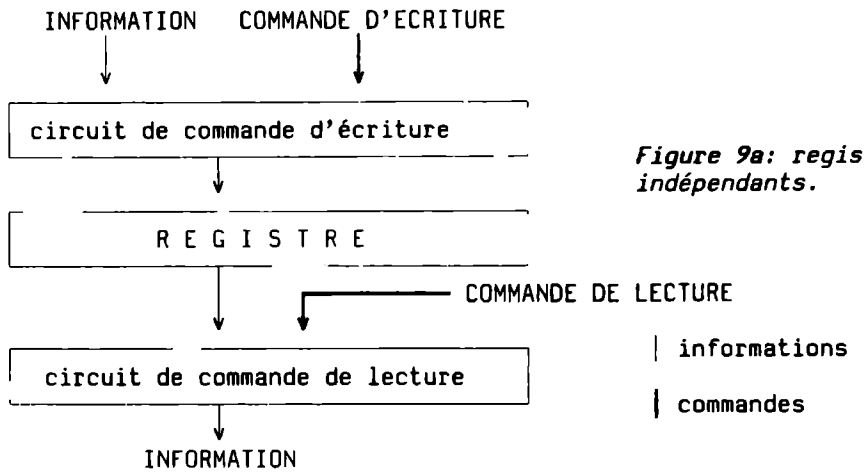


Figure 9a: registres indépendants.

Un tel registre forme donc un circuit séquentiel représenté sous forme de boîte noire figure 9b. Cette organisation est très puissante car chaque registre fonctionne de façon indépendante des autres. Elle est par contre coûteuse puisque chaque registre dispose de ses propres circuits de commande et qu'il faut deux signaux de commande par registre;

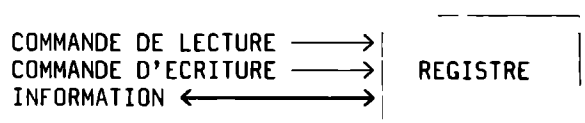


Figure 9b: modèle d'un registre.

- les registres organisés en **mémoire locale**: chaque registre est en fait une case d'une petite mémoire contenue dans le processeur central (local memory, scratchpad en américain). On trouve cette organisation dans la plupart des microprocesseurs. Elle est plus simple et moins coûteuse. Les performances obtenues sont moindres a priori, mais dans ces processeurs, la possibilité d'utiliser plusieurs registres simultanément est faible.
  Le 4004 d'INTEL a ainsi ses 16 registres de 4 bits organisés en mémoire locale. Sur le 4040 on passe à deux mémoires locales (deux "bancs"), la seconde incluant 8 nouveaux registres. La performance est accrue puisque l'on peut simultanément accéder à deux registres, un dans chaque mémoire locale;

- les **pseudo registres**: l'architecture externe manipule des registres qui n'ont aucune existence physique à l'intérieur du processeur central car ils sont simulés par des cases de la mémoire centrale. C'est le cas des microprocesseurs TMS99000 de Texas Instruments.

Mentionnons ici le cas inverse où les registres existent mais peuvent être manipulés comme des cases de la MC car ils correspondent à des adresses MC réservées. Ceci permet de disposer de registres, sans instructions qui leur soient spéciales, et il est ainsi possible de faire bénéficier les registres des instructions opérant sur la mémoire centrale. On trouve cette structure sur les Sperry 1100, la série 16 de Bull, les PRIME en mode R.

9.2.1 – Classification des registres.

Nous avons déjà opéré une première classification des registres au niveau de l'architecture externe, mais l'architecture interne introduit d'autres registres:

- les registres d'**historique** ont pour but d'aider au diagnostic des incidents. Sur les DPS8 de Bull ils contiennent ainsi les n dernières instructions exécutées, sous forme de mémoire circulaire. Dans d'autres processeurs (certains modèles 1100 de Sperry) on conserve les n dernières instructions de branchement;
- les registres de **traduction d'adresse** que nous verrons au Tome II;
- les registres de **contrôle** (CRi) que l'on trouve sur les 370 (et non sur les 360) pour prendre en compte la pagination et les extensions d'architecture.

9.2.2 – Jeux multiples de registres.

Afin d'obtenir des performances plus importantes, les registres sont dupliqués dans certains processeurs. C'est le cas du Z-80 par rapport au 8080. C'est aussi le cas des modèles haut de gamme des 370 où un jeu est réservé aux programmes des utilisateurs et l'autre au système d'exploitation. Les minis orientés vers le traitement en temps réel disposent fréquemment de jeux multiples: les CLASSIC de MODCOMP ont ainsi 16 jeux de registres.

9.3 – Les opérateurs.

Nous avons déjà introduit deux opérateurs dans le processeur. L'opérateur arithmétique et logique (UAL) représenté figure 9c est chargé de réaliser les opérations que nous avons qualifiées d'arithmétiques et de logiques. Il élabore un résultat et positionne en conséquence les indicateurs.

Quant au second opérateur déjà introduit, il s'agit du circuit d'incrémentation du compteur ordinal CO.

Il peut exister d'autres opérateurs lorsque l'on veut améliorer les performances (la puissance de calcul). Nous avons évoqué les options "opérateur flottant" (on parle aussi de flottant câblé) dans le cas du VAX, du MINI6 (l'option s'appelle alors SIP comme Scientific Instruction Package) et de la plupart des mini-ordinateurs de ce type.

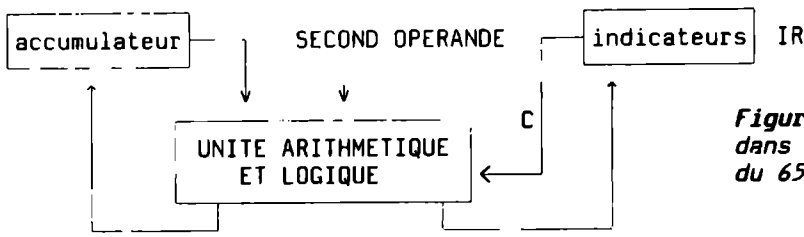


Figure 9c: UAL dans le contexte du 6502.

Les instructions de calcul en décimal et de traitement de chaînes de caractères font aussi l'objet d'option: CIP (Commercial Instruction Package) sur MINI6 de Bull (appelé maintenant DPS6); les 6000 commercialisés par Bull avaient aussi ces instructions (dites EIS comme Extended Instruction Set) en option, elles sont devenues standard sur les 66. Le DPS88 dispose lui de 6 opérateurs:
- le BOPS pour la majorité des opérations binaires sur les entiers, les opérations booléennes, les comparaisons entières, les chargements de registre et les décalages;
- le CEU pour les transferts, les indicateurs visibles du logiciel, les registres adresse, les fonctions de servitude;
- le BINAU pour les multiplication et division entières, le calcul en flottant;
- le DECCU pour le calcul en décimal, les traitements de chaîne et de bits;
- le VMSM pour la gestion de la segmentation et les mécanismes de protection;
- le collecteur pour les opérations de stockage.

Les deux derniers "opérateurs" n'en sont d'ailleurs pas puisqu'ils correspondent simplement à des circuits spécialisés dans l'exécution d'instructions particulières non fonctionnelles.

Le S1000 de NEC, un compatible haut de gamme des DPS8, commercialisé sous le nom de DPS90 par Bull et Honeywell, dispose d'opérateurs vectoriels internes. On peut aussi ajouter sur un ordinateur un opérateur vectoriel quelconque, mais en dehors du sous-système central, en tant que sous-système périphérique auquel le sous-système central envoie des instructions ou des programmes à exécuter.

C'est, entre autres, le cas des opérateurs (array processeur) de Floating Point qui se connectent sur la plupart des ordinateurs actuels. Cette société avait démarré en produisant des opérateurs flottants pour l'IBM 1401.

Dans le domaine des calculateurs scientifiques, le nombre d'opérateurs augmente énormément, on en trouve au moins neuf sur un 7700 de CDC, les opérateurs:
- booléen,
- de normalisation,
- d'addition en entier long,
- de division en flottant,
- de décrémentation,
- de décalage,
- d'addition en flottant,
- de multiplication en flottant,
- de "comptage de population", c'est-à-dire de nombre de bits à un.

Le CRAY 1 y ajoute des opérateurs manipulant des vecteurs de 64 mots de 64 bits: additionneur, opérateur de décalage et opérateur logique.

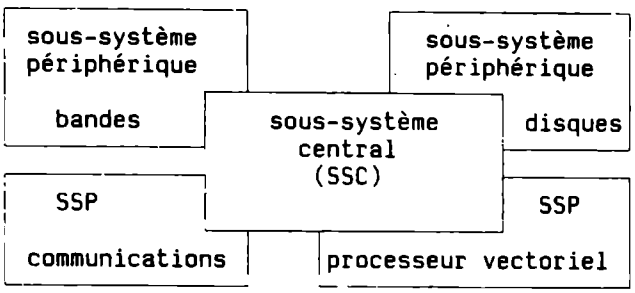


Figure 9d: décomposition de l'ordinateur en sous-systèmes, l'un d'eux étant un opérateur spécialisé.

9.3.1 – Coprocesseurs.

Ce terme de coprocesseur dépasse le cadre des opérateurs puisque l'on y parle de "processeur" associé. Il est cependant utilisé dans le contexte des opérateurs car les coprocesseurs arithmétiques (on dit aussi numériques) complètent les possibilités de calcul des microprocesseurs dans le domaine du flottant, et parfois dans celui des entiers et même du décimal (cas du 8087). Ils sont appelés coprocesseurs parce que, comme un processeur, ils disposent de leur propre jeu d'instructions, de leurs propres registres, et on peut voir des ordinateurs avec un processeur 68000 de Motorola et un coprocesseur de NS (cas de TELMAT avec le NS 32081 sur un SM90).

Ils correspondent donc à une solution intermédiaire entre l'option "flottant câblé" qui trouve sa place dans l'architecture externe du processeur et le processeur de tableaux (array processor) périphérique qui est totalement étranger à cette architecture externe dans laquelle il s'intègre en tant que sous-système périphérique.

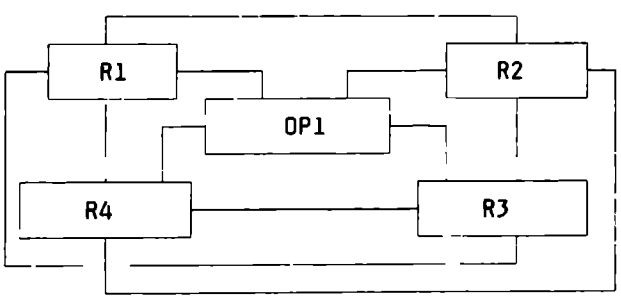


Figure 9e: interconnexions directes entre 4 registres et 1 opérateur.

9.4 – Interconnexion des registres, des opérateurs.

On distingue deux grands modes d'interconnexion entre les différents circuits qui composent le processeur:

- les connexions directes représentées figure 9e. Cette solution offre les performances maximales puisque l'on peut relier tout couple et

avoir des transferts simultanés, mais le coût en est élevé car il faut n(n-1)/2 liaisons pour n éléments;
- les interconnexions organisées autour d'un **bus** offrent un coût moindre, au détriment des performances, puisqu'il n'y a plus qu'un seul transfert à la fois, même si l'on imagine le cas où un même registre diffuserait son contenu vers plusieurs autres registres et opérateurs.

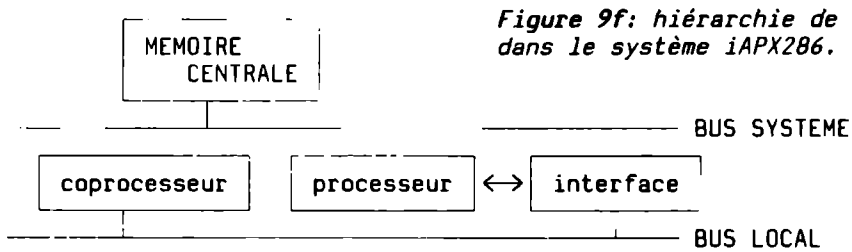


Figure 9f: hiérarchie de bus dans le système iAPX286.

9.4.1 – Hiérarchie de bus.

Si le bus séquentialise les transferts dans la mesure où il supprime un parallélisme potentiel, il est cependant possible d'en utiliser plusieurs:
- nous verrons que de nombreux microprocesseurs ont deux bus distincts pour transférer les données et les adresses,
- le coprocesseur numérique 80287 d'INTEL dispose en interne d'un bus pour l'exposant et d'un bus pour la mantisse, comme le montre la figure 9g.
- des bus de niveaux différents comme les bus local et système de l'iAPX286 schématisés figure 9f.

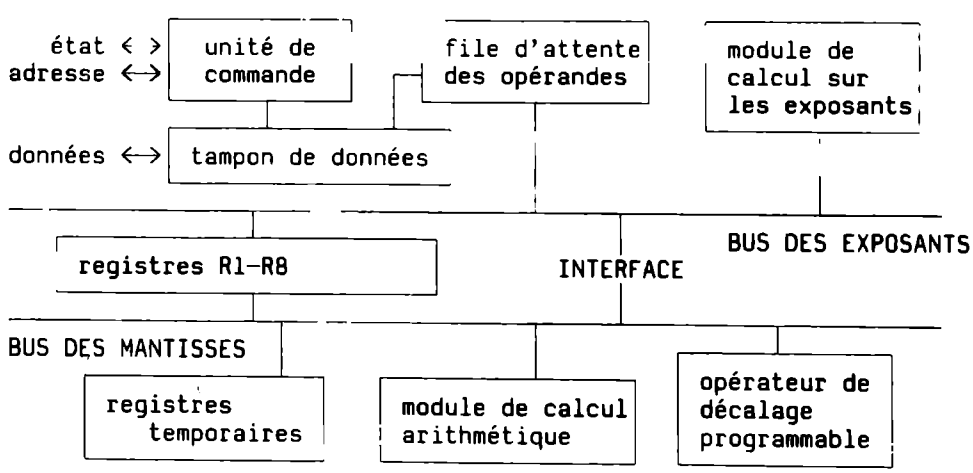


Figure 9g: les deux bus internes du coprocesseur 80287.

9.4.2 – Machine illustrant les notions présentées.

Afin d'illustrer plus concrètement les notions présentées dans les chapitres suivants, nous utiliserons un sous-système central appelé MODEL1 (figure 9h).

Son architecture externe est celle du 6502. Au niveau de l'architecture interne il dispose:

- d'un bus BUS.D de huit bits de large pour transférer les données;
- d'un bus BUS.A de seize bits de large pour transférer les adresses;
- d'un bus de commande BUS.C que nous détaillerons au fur et à mesure des besoins avec des signaux appelés BUS.CX;
- d'un opérateur d'incrémentation ("+1" dans la figure 9h) sur 16 bits pour le compteur ordinal CO;
- d'une unité arithmétique et logique sur 8 bits avec trois entrées potentielles: l'accumulateur, le bus BUS.D, l'indicateur C;
- d'un cycle de base qui lui permet de réaliser un transfert entre deux registres en un seul cycle;
- l'accumulateur est, dans un même cycle, entrée de l'UAL puis destination du résultat.

Ces termes de cycle seront expliqués dans le chapitre suivant.

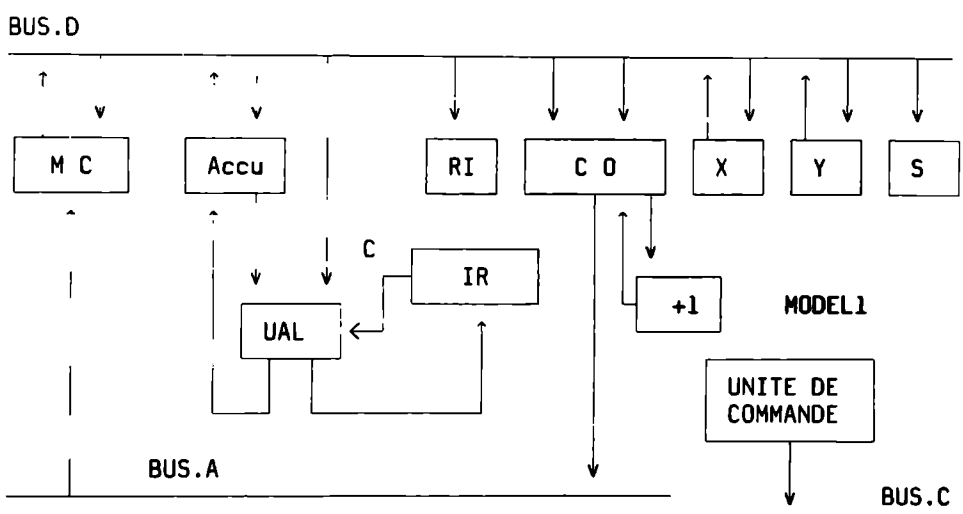


Figure 9h: structure de la machine MODEL1.

chapitre 10

unité de commande et machine câblée

10.1 – L'unité de commande.

Nous avons vu jusqu'ici tous les composants de base du sous-système central, mais sans parler de celui qui les pilote tous. Les opérateurs sont là pour réaliser des opérations, les registres pour stocker les informations, les bus pour transporter les informations entre les circuits, mais ce sont essentiellement des éléments **passifs** qui ne se mettent en oeuvre que lorsqu'ils en reçoivent l'ordre. Celui qui délivre ces ordres s'appelle **l'unité de commande.**

L'unité de commande est un circuit séquentiel qui, en fonction de l'état du processeur, fournit les signaux de commande qui mettront en oeuvre les différents éléments du chemin des données qui doivent être activés (figure 10a).

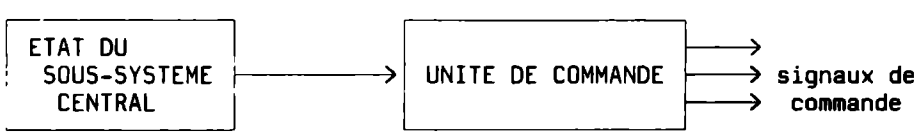


Figure 10a: rôle de l'unité de commande.

10.2 – Etat du sous-système central.

Diverses informations composent l'état sur lequel se base l'unité de commande pour délivrer ses ordres:

- le compteur ordinal (CO),
- le contenu du registre instruction RI: il est bien évident que la partie du chemin de données qui sera mise en oeuvre dépend de l'instruction à exécuter. Dans RI, l'unité de commande prendra en compte:
  * le code opération afin de connaître par exemple la fonction que devra assurer l'unité arithmétique et logique (UAL),
  * le mode d'adressage afin d'activer correctement le circuit de transformation des adresses s'il y a lieu,
- les indicateurs stockés dans le registre IR,

- les indicateurs d'état des opérateurs, de la mémoire centrale, pour savoir s'ils sont libres par exemple,
- le mode de fonctionnement, privilégié ou non,
- certaines informations que nous n'avons pas introduites jusqu'ici.

A cet état qui se situe au niveau de l'instruction (il ne change que lorsque l'on change d'instruction) l'unité de commande ajoute son propre état qui lui permet de distinguer plusieurs sous-états à l'intérieur de l'exécution d'une instruction (figure 10b).

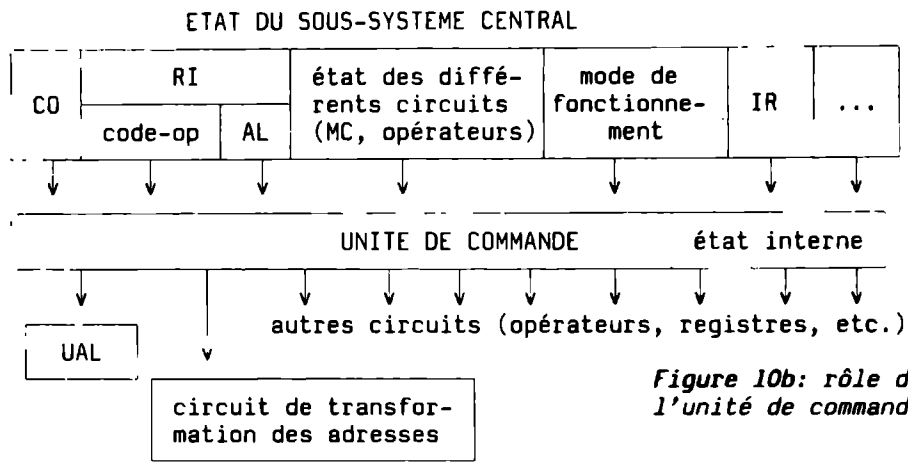


Figure 10b: rôle de l'unité de commande.

10.3 – Les phases de l'exécution d'une instruction.

L'exécution des instructions est découpée en phases successives dans le temps. L'objectif poursuivi est de regrouper ce qui dans l'exécution d'un groupe d'instructions est identique pour toutes ces instructions. Le gain est évident: si une fonction est commune à plusieurs instructions, elle ne sera ainsi réalisée qu'une seule fois.

Or il y a une séquence d'actions élémentaires qui est commune à toutes les instructions et que l'on appelle **phase de chargement** ou phase d'appel. Elle consiste à aller chercher l'instruction en mémoire centrale pour la ranger dans le registre instruction RI.

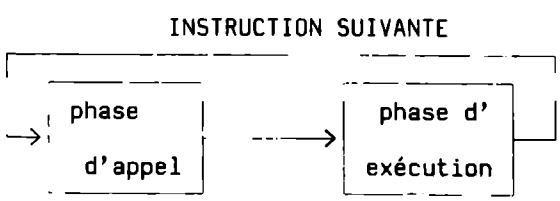


Figure 10c: décomposition en phases de l'exécution des instructions.

Comme toutes les instructions ont une telle phase d'appel (fetch en américain), le reste du temps consacré à leur exécution formera la seconde phase appelée **phase d'exécution**. Le processeur ne fait donc qu'enchaîner des instructions en suivant ces deux phases, comme le montre la figure 10c, à la façon d'un moteur à deux temps.

Le découpage en deux phases ne constitue pas une limite, et certains ordinateurs décomposent en trois, quatre phases, voire plus. Nous verrons dans le chapitre 17 que ces phases peuvent d'ailleurs se dérouler en parallèle afin d'augmenter la vitesse de traitement. Mais pour l'instant nous restons avec un processeur aussi simple que possible, c'est-à-dire très séquentiel, les actions se déroulant les unes après les autres.

PHASE D'APPEL	PHASE D'EXECUTION
. lire l'instruction à l'adresse contenue dans CO . ranger l'instruction dans RI . incrémenter CO	. rechercher les opérandes selon les AL . réaliser l'opération définie dans RI

Figure 10d: fonctions des différentes phases.

10.4 – Processeurs synchrones et asynchrones. Cycles.

Les signaux de commande (ou **microcommandes**) envoyés par l'unité de commande sont enchaînés de façon:

- **arythmique**: les différents éléments du sous-système central envoient des signaux dès qu'ils ont réalisé la fonction définie par la dernière microcommande reçue. L'unité de commande peut alors envoyer les microcommandes qui attendaient tel signal en provenance de tel élément. On a affaire à un calculateur **asynchrone** (asynchronous computer en américain). Rares sont les ordinateurs de ce type. Les DPS8 de Bull en font partie, mais ce n'est plus le cas des DPS88 et DPS90. On ne pourra pas parler de cycle de base ou de cycle machine: ils n'existent pas;
- **rythmique**: l'unité de commande est alors rythmée par une horloge et les envois de microcommandes sont faits lors du top de l'horloge (front montant ou descendant) avec ou sans retard à l'intérieur du cycle. On a alors affaire à des processeurs **synchrones**. La grande majorité des ordinateurs fait partie de cette catégorie. La vitesse de l'horloge se mesure en mégahertz (MHz): le 6502 a généralement une horloge à 2 MHz, donc un cycle de 500 ns (nanosecondes); les Z-80 de haut de gamme "tournent" à 8 MHz en 1985, soit un cycle de 125 ns. Le CRAY 2 a un cycle de 4,1 ns.

  La période de l'horloge est définie en considérant les transferts élémentaires appelés **microtransferts** que doit réaliser le processeur. Ces microtransferts représentent le transfert entre deux registres du processeur, en passant éventuellement à travers un opérateur simple. Il faut distinguer deux cas:
  * le cas des microtransferts rapides qui s'effectuent en une période d'horloge, ou en un cycle,
  * le cas des microtransferts faisant intervenir une mémoire plus lente que les registres, des opérateurs complexes, trop longs pour exécuter leur fonction en un seul cycle. L'unité de commande:

. soit alloue un nombre de cycles que le microtransfert ne doit pas dépasser,
. soit attend que l'élément le plus lent lui signale qu'il en a terminé. On retrouve le mécanisme de base des processeurs asynchrones et on parlera alors **d'asynchronisme partiel** ou d'allongement du cycle.
Le signal RDY du 6502 permet à un circuit extérieur d'allonger ainsi le cycle du processeur.

L'horloge interne distribuant les cycles est apparue dès 1944 où on la trouve sur une machine développée par l'université de Harvard et IBM. Il existe des systèmes avec plusieurs horloges (systèmes biphasés, polyphasés) ainsi que des circuits endochrones, cas particulier d'asynchronisme local (voir [MEA83]).

En résumé, nous venons de voir que l'instruction qui forme un tout insécable au niveau de l'architecture externe, se décompose en phases au niveau de l'architecture interne. Ces phases comprennent plusieurs cycles successifs dans le cas, très répandu, des processeurs synchrones. Chacun de ces cycles correspond à un ou plusieurs microtransferts qui se déroulent en parallèle.

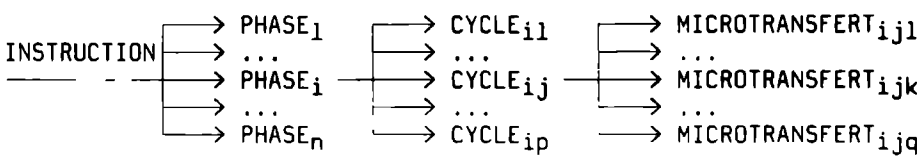


Figure 10e: décomposition de l'exécution en phases, cycles et microtransferts.

10.5 – Séquenceur câblé d'un processeur synchrone.

Le circuit principal de l'unité de commande est le **séquenceur**. Il existe deux types de séquenceur:
- le séquenceur **câblé** comportant des compteurs pour cadencer les étapes successives nécessaires au déroulement complet de chaque instruction, ainsi qu'un circuit combinatoire pour générer les microcommandes des différents cycles;
- le séquenceur **microprogrammé** qui possède une mémoire interne contenant un petit programme appelé **microprogramme** (microprogram en américain) pour chacune des instructions.

Le séquenceur câblé comprend:
- un **compteur de phases** (CP), modulo 2 dans notre cas puisque nous prendrons le processeur défini au chapitre précédent avec deux phases seulement. Ce compteur est doté d'un opérateur d'incrémentation et d'un dispositif de remise à zéro;
- un **compteur de cycles** (CC), modulo le nombre maximum de cycles que compte la phase la plus longue. Il est doté d'un opérateur d'incrémentation et d'un dispositif de remise à zéro;
- un **circuit combinatoire** dont les entrées sont l'état de l'ordinateur et les deux compteurs CP et CC.

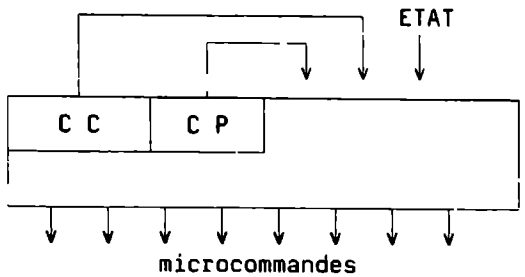


Figure 10f: structure d'un séquenceur câblé.

Les sorties sont les microcommandes qui permettent:
* d'ouvrir les portes de sortie des registres, c'est-à-dire de lire le contenu de ces registres,
* d'ouvrir les portes d'entrée de ces registres afin d'y ranger une nouvelle information. Un microtransfert minimum comprend en effet:
  . l'envoi de la microcommande ouvrant les portes de sortie du registre source,
  . l'envoi de la microcommande ouvrant les portes d'entrée du registre destination (figure 10g).
* d'indiquer à l'UAL l'opération à exécuter (addition, soustraction, décalage, etc.),
* de commander les signaux sur les bus de commande.

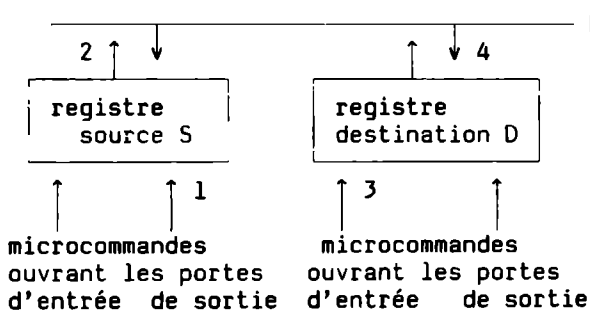


Figure 10g: microcommandes envoyées lors d'un microtransfert.

1 ouverture de S
2 transfert de S sur BUS
3 ouverture de D
4 transfert de BUS dans D

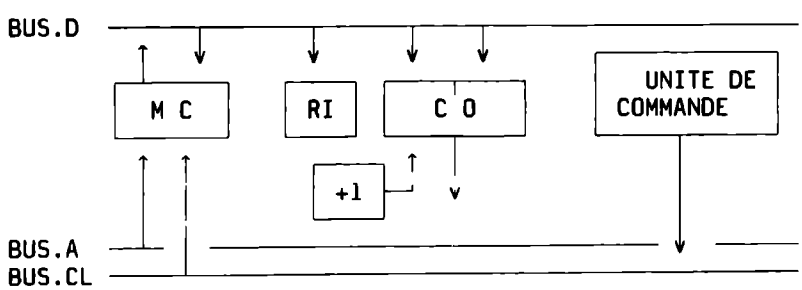


Figure 10h: structure de la machine MODEL1 utilisée pendant la phase d'appel (BUS.CL: commande lecture).

10.6 – Exemples de cycles et de microcommandes.

Nous donnons dans cette section quelques exemples qui permettent de voir à quel niveau de détail les concepteurs de la structure interne doivent descendre.

Nous supposerons, sauf indication contraire, que la mémoire centrale a un cycle (appelé majeur) plus court que le cycle du processeur. Il en est fréquemment ainsi dans les microprocesseurs, mais non dans les minis, moyens ou gros ordinateurs.

Nous supposons de plus que la mémoire centrale est libre en début de phase d'appel.

10.6.1 – Phase d'appel.

CP	CC	MICROCOMMANDES	COMMENTAIRES
0	0	BUS.CL ⟵ lecture MC	ordre envoyé à MC
		BUS.A ⟵ (CO)	adresse à lire
		RI ⟵ (BUS.D)	avec le retard voulu
		CO ⟵ (CO)+1	incrémentation de CO
		CP ⟵ (CP)+1	passage à la phase suivante

Tableau 10i: phase d'appel dans le cas de MODEL1.

Le chronogramme des signaux de la phase d'appel donnée en figure 10i se trouve figure 10j.

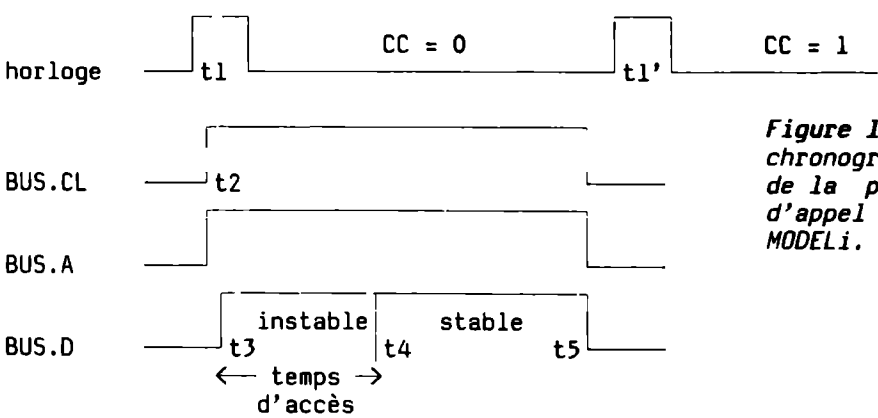


Figure 10j: chronogramme de la phase d'appel de MODELi.

Les choses sont plus complexes dans la réalité, les signaux y sont moins verticaux, moins alignés et un spécialiste y trouverait à redire, tiendrait à introduire les logiques polyphasées. Mais seuls les principes nous intéressent ici et nous devons nous en tenir à un certain niveau d'abstraction, ainsi que nous l'avons exposé dans l'in-

troduction. On lira avec intérêt une comparaison entre les accès mémoire de l'iAPX286 et du 68000 dans [INT85A].

L'unique cycle qui nous intéresse ici commence à l'instant t1. Dès que la mémoire détecte BUS.CL (cf. figure 10h) et que l'adresse BUS.A devient stable (instant t2), elle peut commander le bus de données BUS.D. Lorsque le mot demandé par le processeur est trouvé par la mémoire centrale, après un temps appelé temps d'accès de la mémoire, il est placé (instant t4) sur BUS.D qui devient stable, et le processeur peut transférer son contenu dans RI (rôle de la microcommande "RI <-- (BUS.D)"). Soit l'unité de commande connaît t4 par construction, soit la mémoire positionne un signal "PRET" que nous illustrerons dans d'autres exemples.

Si nous supposons maintenant que la mémoire a un temps d'accès égal à deux fois le cycle de base (MODEL2), nous aurons la phase d'appel décrite dans les figures 10k et 10l.

CP	CC	MICROCOMMANDES	COMMENTAIRES
0	0	BUS.CL ⟵ lecture MC BUS.A ⟵ (CO) CC ⟵ (CC)+1	ordre envoyé à MC adresse à lire cycle suivant
	1	CC ⟵ (CC)+1	attente de MC
	2	BUS.D ⟵ (MC) RI ⟵ (BUS.D) CO ⟵ (CO)+1 CP ⟵ (CP)+1 CC ⟵ 0	instruction sur le bus instruction dans RI incrémentation CO passage à la phase suivante

Tableau 10k: phase d'appel dans le cas de MODEL2.

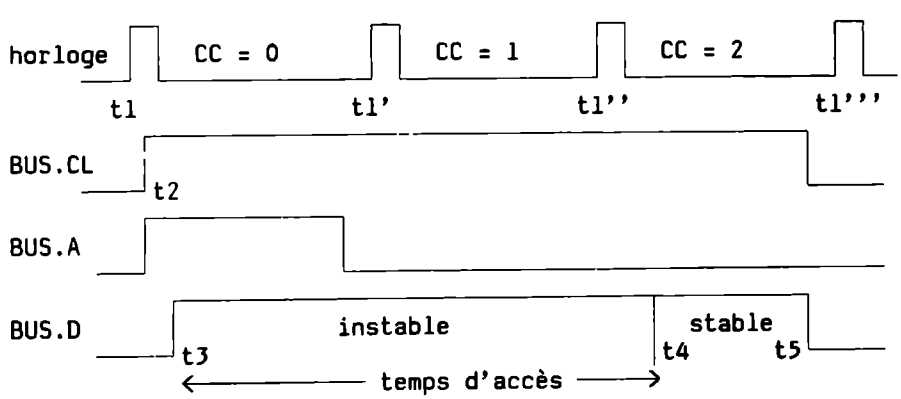


Figure 10l: chronogramme de la phase d'appel de MODEL2.

Si nous supposons maintenant que le processeur ne connaît pas le temps d'accès de la mémoire centrale, nous arrivons à une situation décrite avec MODEL3 dans les figures 10m et 10o. La mémoire centrale y est vue du processeur sous forme de deux registres:
- **RAM**, le registre adresse mémoire qui contient l'adresse de la case que l'on veut lire, ou dans laquelle on veut écrire;
- le registre tampon mémoire **RM** qui contient la valeur que l'on veut écrire ou lire.

Nous ajoutons le signal BUS.CM sur le bus de commande afin de connaître l'état du registre tampon RM de la mémoire (figure 10p):
- quand BUS.CM est bas le registre RM ne contient rien de significatif du fait de la mémoire,
- quand il est haut il contient l'information que l'on veut lire.

La mémoire centrale est toujours supposée libre en début de phase d'appel.

CP	CC	MICROCOMMANDES	COMMENTAIRES
0	0	BUS.A ⟵ (CO) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	1	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	2	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	3	BUS.D ⟵ (RM) RI ⟵ (BUS.D) CO ⟵ (CO)+1 CP ⟵ (CP)+1 CC ⟵ 0	instruction lue de RM dans RI incrémentation CO passage à la phase suivante

Tableau 10m: phase d'appel dans le cas de MODEL3.

On voit apparaître une microcommande conditionnelle, "CC ⟵ (CC)+1 si BUS.CM haut": elle représente une sortie d'un opérateur ET qui a comme entrées la microcommande "CC ⟵ (CC)+1" et le signal BUS.CM (figure 10n).

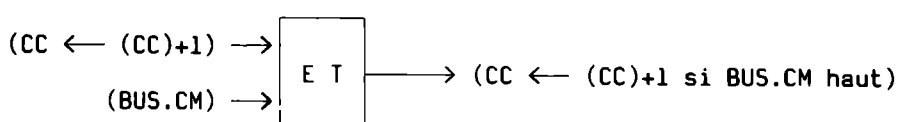


Figure 10n: microcommande conditionnelle.

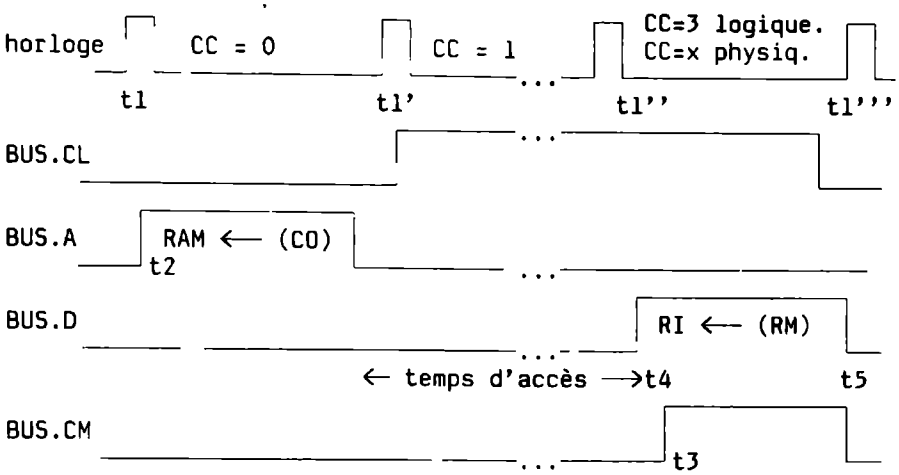


Figure 10o: chronogramme de la phase d'appel de MODEL3.

Dans l'exemple de MODEL3 la mémoire fonctionne de façon asynchrone par rapport au processeur central. La seule hypothèse de synchronisation faite concerne la disponibilité de la mémoire en début de phase d'appel. Si elle n'est pas vérifiée, il faudra ajouter un cycle en début de phase d'appel ou conditionner les microcommandes par un test sur l'état de la mémoire. On trouvera dans [MIT84] le dialogue processeur-mémoire dans l'AMD29116.

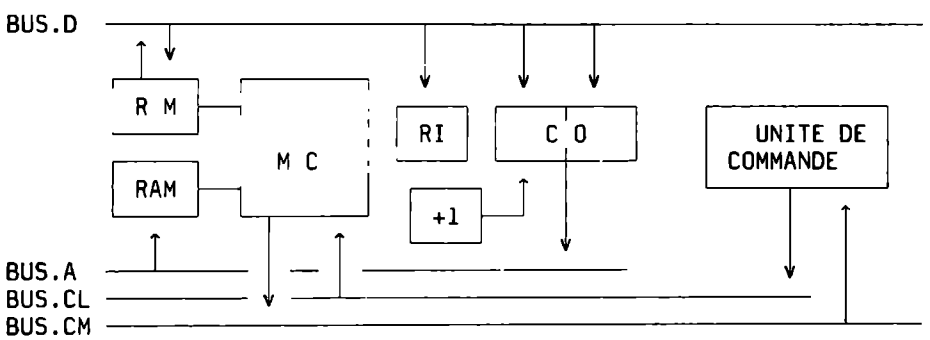


Figure 10p: structure de la machine MODEL3 utilisée pendant la phase d'appel.

10.6.2 – Exécution de LDA# avec MODEL1.

LDA# charge dans l'accumulateur l'octet qui est rangé en mémoire après le code opération. La phase d'exécution de LDA# est analogue à sa phase de chargement. L'exécution complète de LDA# nécessite ainsi deux cycles, ce qu'indique la documentation du concepteur du 6502.

CP	CC	MICROCOMMANDES	COMMENTAIRES
1	0	BUS.CL ⟵ lecture MC	ordre envoyé à MC
		BUS.A ⟵ (CO)	adresse à lire
		A ⟵ (BUS.D)	avec le retard voulu chargement dans Acc
		CO ⟵ (CO)+1	incrémentation de CO
		CP ⟵ 0	passage à la phase d'appel suivante

Tableau 10q: phase d'exécution dans le cas de MODEL1.

10.6.3 – Exécution de LDAA avec MODEL3.

Lorsque l'on entame la phase d'exécution de LDAA, la mémoire et CO ont un contenu résumé figure 10r.

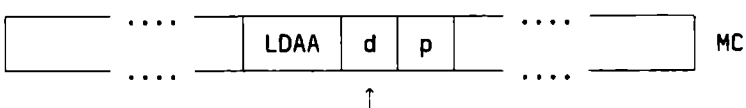


Figure 10r: CO pointe vers le premier octet de l'adresse logique en début de phase d'exécution.

p : numéro de page 6502
d : déplacement dans la page

La structure actuelle de MODEL3 ne permet pas de réaliser l'exécution de cette instruction qui nécessite de faire passer deux octets de la mémoire centrale dans le registre adresse mémoire RAM, sans modifier le contenu des registres actuels. Or ceux-ci soit ne peuvent pas être modifiés (cas de CO), soit sont visibles au niveau de l'architecture externe et doivent donc conserver le même contenu avant et après exécution, sauf bien sûr si l'instruction les modifie du fait de son code opération.

Nous introduirons donc un registre supplémentaire, transparent pour l'utilisateur, RINT1 (registre interne numéro 1) de 16 bits, relié aux bus adresse et données comme indiqué figure 10s.

Notons, tableau 10t, que le compteur ordinal n'est pas incrémenté à la suite de la lecture du cycle numéro 9: on se déplace alors dans les données, et non pas dans les instructions.

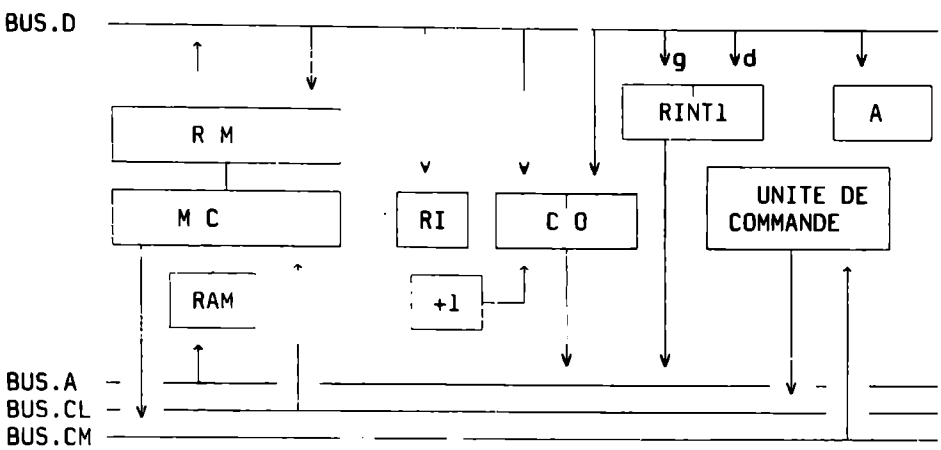


Figure 10s: structure de la machine MODEL3 utilisée pendant la phase d'exécution de LDAA.

10.6.4 – Phase d'exécution de ADCA avec MODEL3.

L'exécution de ADCA est identique à celle de LDAA, à l'exception du cycle 11 dans le tableau 10u. Dans ce cycle 11 du tableau 10t, l'UAL reçoit le code opération (addition) du circuit de décodage du registre d'instruction (cf. figures 11f et 11g).

On constate, dans le cas de MODEL3, qu'il faut 16 cycles (4 dans la phase d'appel, 12 dans la phase d'exécution) pour exécuter l'addition qui n'apparaît réellement que dans le seul cycle [1,11].

CP	CC	MICROCOMMANDES	COMMENTAIRES
1	11	BUS.D⟵(RM)	opérande de ADCA à
		UAL ⟵(BUS.D)	l'entrée de UAL
		UAL ⟵(A)	accumulat. sur l'autre
			entrée de l'UAL,
		UAL ⟵(C)	le report aussi
		A ⟵(UAL)	résultat dans A
		UAL ⟵(RI)	RI définit l'opération
		C ⟵(UAL)	les indicateurs C, Z,
		Z ⟵(UAL)	N et O sont posi-
		N ⟵(UAL)	tionnés
		O ⟵(UAL)	
		CC ⟵0	phase d'appel
		CP ⟵0	suivante

Tableau 10t: cycle 11 de la phase d'exécution de ADCA.

CP	CC	MICROCOMMANDES	COMMENTAIRES
1	0	BUS.A ⟵ (CO) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	1	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	2	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	3	BUS.D ⟵ (RM) RINT1d ⟵ (BUS.D) CO ⟵ (CO)+1 CC ⟵ (CC)+1	déplacement lu dans RINT1d incrémentation CO cycle suivant
	4	BUS.A ⟵ (CO) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	5	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	6	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	7	BUS.D ⟵ (RM) RINT1g ⟵ (BUS.D) CO ⟵ (CO)+1 CC ⟵ (CC)+1	numéro de page lu dans RINT1g incrémentation CO cycle suivant
	8	BUS.A ⟵ (RINT1) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	9	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	10	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	11	BUS.D ⟵ (RM) A ⟵ (BUS.D) CP ⟵ 0 CC ⟵ 0	chargement de A passage à la phase d'appel suivante

Tableau 10u: phase d'exécution de LDAA dans le cas de MODEL3.

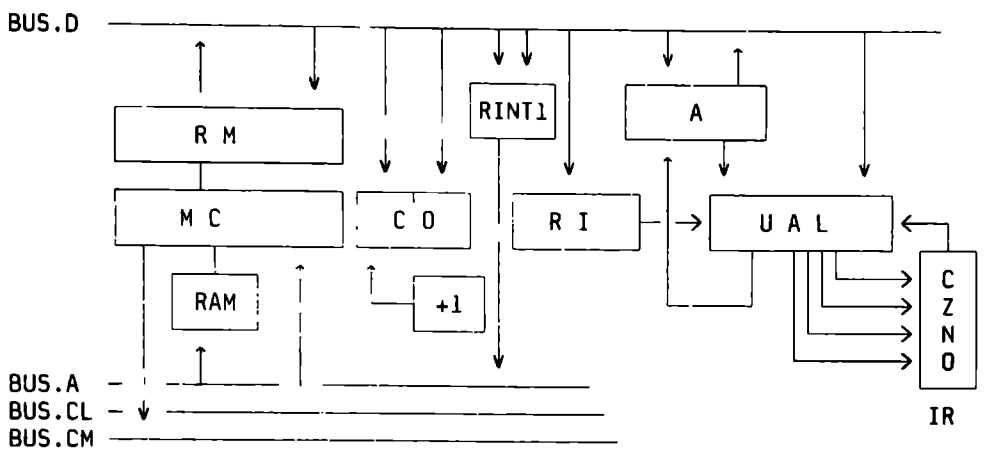


Figure 10v: schéma de MODEL3 correspondant à l'exécution de ADCA.

10.6.5 – Exécution de INCA avec MODEL3.

Cette phase est identique à celle de LDAA, jusqu'au cycle 10 de la phase 1 inclus. Au moment d'exécuter le cycle 11 se posent deux problèmes:

- l'incrémentation nécessite d'utiliser l'UAL, mais actuellement celle-ci ne sait ranger le résultat que dans l'accumulateur A. Or INC incrémente un mot mémoire sans modifier l'accumulateur. Il faut donc compléter la structure interne de MODEL3 pour arriver à MODEL4: en ajoutant un registre RINT2 (registre interne numéro 2) de 8 bits qui peut recevoir le résultat de l'UAL (figure 10w);

- le second problème concerne l'état de la mémoire centrale. Nous avons supposé dans les exemples précédents que la mémoire était toujours disponible lorsque l'unité de commande lançait un nouvel ordre de lecture. Il n'en est pas toujours ainsi car temps d'accès et temps de cycle pour une mémoire sont très souvent différents:
  * le **temps d'accès** joue un rôle dans le cas de la lecture, il donne le temps qui s'écoule entre le moment où on envoie l'ordre de lecture et celui où l'on peut disposer des données lues,
  * le **temps de cycle** est le temps pendant lequel la mémoire est occupée par un accès, que ce soit de lecture ou d'écriture.

  Deux lectures ne peuvent donc être lancées à moins d'un cycle de la mémoire centrale (cycle dit majeur). Pour que le processeur tienne compte de l'état de la mémoire centrale, nous introduirons un nouveau signal sur le bus de contrôle, BUS.CP, qui est mis à l'état haut quand la mémoire est libre, prête pour un nouvel accès.

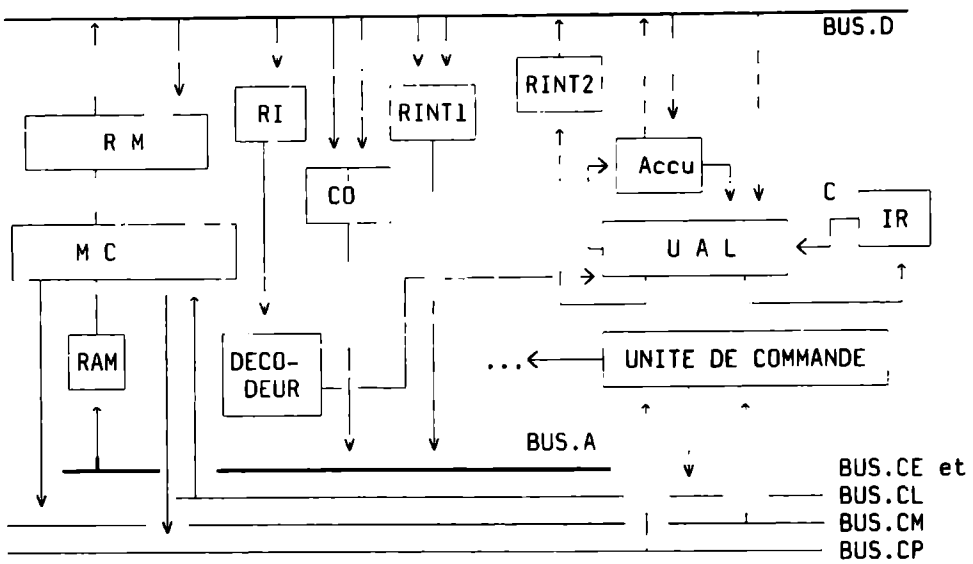


Figure 10w: MODEL4, partie utilisée pour INCA.

La phase d'exécution de INCA est donnée dans le tableau 10y. Nous aboutissons à un résultat peu optimal, surtout si l'on compare avec le nombre de cycles donné par le constructeur. Cela est dû à l'asynchronisme entre la mémoire et le processeur.

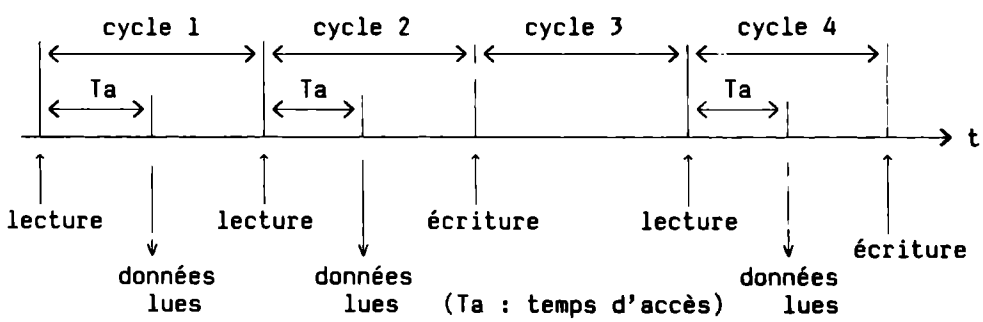


Figure 10x: séquence de lectures/écritures (cycle i = un cycle de mémoire centrale).

On notera dans le cycle 14 que INCA ne positionne que les indicateurs Z et N; C et O conservent eux le souvenir de la dernière opération concernant l'accumulateur A. Ils ne reflètent jamais les opérations en mémoire centrale.

CP	CC	MICROCOMMANDES	COMMENTAIRES
	0	CC ⟵ (CC)+1 si BUS.CP haut	attente MC libre
1	1	BUS.A ⟵ (CO) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	2	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	3	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	4	BUS.D ⟵ (RM) RINTld ⟵ (BUS.D) CO ⟵ (CO)+1 CC ⟵ (CC)+1	déplacement lu dans RINTld incrémentation CO cycle suivant
	5	CC ⟵ (CC)+1 si BUS.CP haut	attente MC libre
	6	BUS.A ⟵ (CO) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	7	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	8	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC
	9	BUS.D ⟵ (RM) RINTlg ⟵ (BUS.D) CO ⟵ (CO)+1 CC ⟵ (CC)+1	numéro de page lu dans RINTlg incrémentation CO cycle suivant
	10	CC ⟵ (CC)+1 si BUS.CP haut	attente MC libre
	11	BUS.A ⟵ (RINTl) RAM ⟵ (BUS.A) CC ⟵ (CC)+1	adresse à lire dans RAM cycle suivant
	12	BUS.CL ⟵ lecture MC CC ⟵ (CC)+1	ordre envoyé à MC
	13	CC ⟵ (CC)+1 si BUS.CM haut	attente de MC

Tableau 10y: exécution de INCA (première partie).

CP	CC	MICROCOMMANDES	COMMENTAIRES
1	14	BUS.D ⟵ (RM) UAL ⟵ (BUS.D) RINT2 ⟵ (UAL) UAL ⟵ (RI) Z ⟵ (UAL) N ⟵ (UAL) CC ⟵ (CC)+1	opérande de INCA à l'entrée de l'UAL résultat dans RINT2 opération déduite du code op dans RI positionnement des indicateurs Z et N cycle suivant
	15	CC ⟵ (CC)+1 si BUS.CP haut	attente MC libre
	16	BUS.D ⟵ (RINT2) RM ⟵ (BUS.D) BUS.CE ⟵ écrire CC ⟵ (CC)+1	écrire RINT2 sans changer d'adresse
	17	CC ⟵ 0 si BUS.CM haut CP ⟵ 0 si BUS.CM haut	préparation de la phase d'appel de l'instruction suivante

Tableau 10y: phase d'exécution de INCA dans le cas de MODEL4.

10.7 – Résumé.

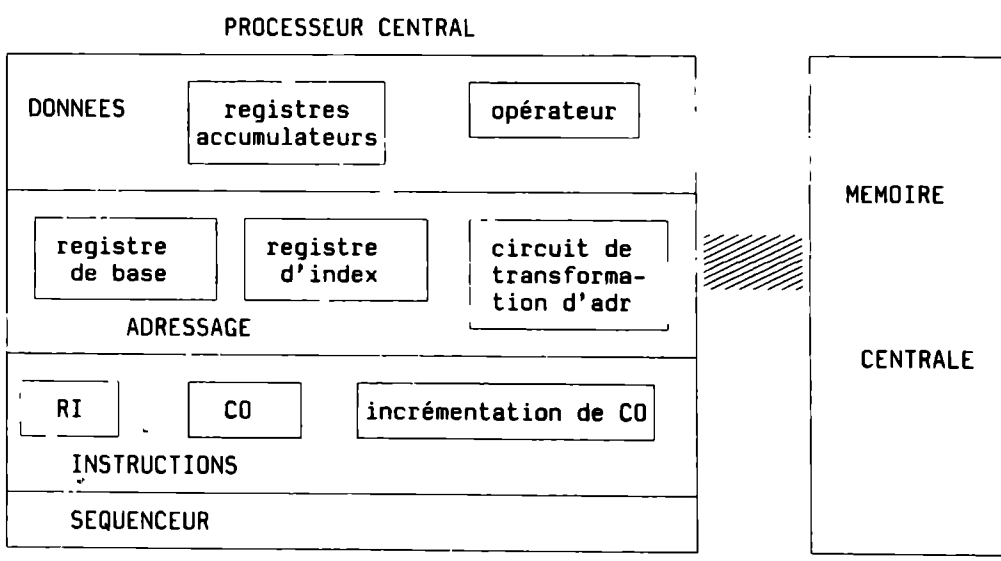


Figure 10z: structure actualisée du sous-système central.

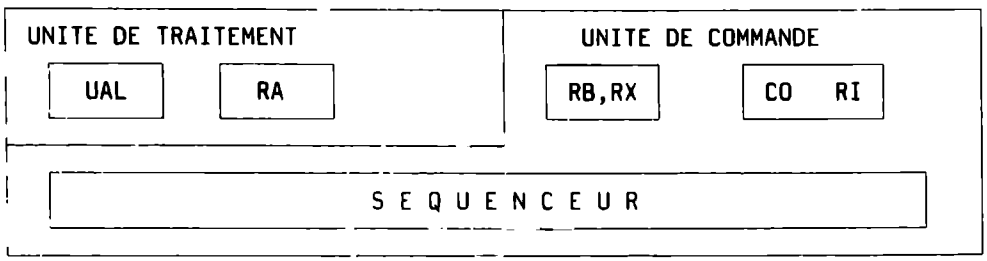


Figure 10aa: *autre vue actualisée du processeur (RA : registre accumulateur).*

chapitre 11

machines microprogrammées

11.1 – Séquenceur microprogrammé.

Nous avons vu dans le chapitre précédent qu'il y avait deux façons différentes de réaliser un séquenceur: par des circuits spécialisés, c'est ce que l'on appelle le séquenceur **câblé**, ou bien par une méthode qui présente des analogies avec la programmation, c'est la solution appelée **microprogrammée**.

Qu'est-ce qu'une solution analogue à la programmation? D'après ce que nous connaissons du sous-système central, c'est une solution qui fait appel à des programmes stockés dans une mémoire, le programme étant constitué d'instructions rangées en séquence et enchaînées les unes après les autres.

C'est ainsi que dans le séquenceur microprogrammé nous trouverons:

- une mémoire interne appelée **mémoire de commande** puisqu'elle est propre à l'unité de commande,
- des **micro-instructions** rangées dans les mots de cette mémoire.

Ces micro-instructions manipuleront des objets qui sont ceux-là même que l'on trouve au niveau de l'architecture interne. Ce sont elles qui généreront les microcommandes que nous avons vues précédemment car le séquenceur n'a pas changé vis-à-vis de l'extérieur. Il est toujours le pilote qui envoie les mêmes microcommandes, que la solution retenue soit câblée ou microprogrammée.

L'ensemble formé par la mémoire de commande, les registres et la logique associée est parfois appelé **micromachine**. Il correspond au CSE (Control Storage Element) de l'IBM 3083 par exemple.

11.2 – Organisation et fonctionnement.

L'exécution de chaque instruction provoque le déroulement d'un certain nombre de micro-instructions formant le microprogramme de cette instruction.

La mémoire de commande est aussi appelée **mémoire de contrôle** (control memory, control store). Le schéma de la figure 11a est un schéma de principe: toutes les micro-instructions de l'interprétation

d'une même instruction ne sont pas toujours consécutives en mémoire de commande et surtout on retrouve dans le cas de la microprogrammation les phases introduites précédemment.

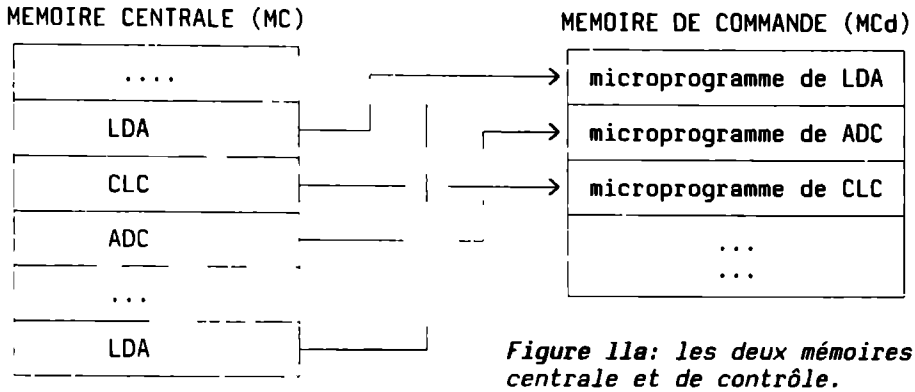


Figure 11a: les deux mémoires, centrale et de contrôle.

Le problème à résoudre étant le même que celui de l'hypothèse câblée, on verra apparaître un microprogramme correspondant à la phase d'appel, et des microprogrammes liés aux phases d'exécution des instructions.

L'adresse du microprogramme d'une phase d'exécution se déduit du code opération qui se trouve dans RI (figure 11b), soit directement (le code opération est égal à l'adresse de la première micro-instruction, ou aux poids forts de cette adresse), soit en passant par un circuit de décodage. [AME85] indique que le codage dense d'un grand nombre d'instructions dans les HP3000 ne permet pas d'obtenir des solutions simples quant au décodage; il y a généralement une table avec une entrée par instruction qui donne l'adresse du microprogramme concerné; dans le HP 3000/64 les 10 bits de gauche de l'instruction sont utilisés pour indexer cette table, solution coûteuse mais justifiée pour un ordinateur de cette puissance; le 3000/37 par contre n'a qu'une table de 256 entrées indexée par les huit bits de gauche de l'instruction, le décodage de l'instruction y prend deux cycles.

A la fin du microprogramme d'une phase d'exécution, on exécute toujours une micro-instruction qui provoque un branchement sur la première micro-instruction de la phase d'appel, et l'enchaînement est assuré.

11.3 – Cycle et micro-instruction.

La microprogrammation est généralement employée avec des processeurs synchrones, ce qui n'est pas surprenant puisqu'elle fait intervenir une mémoire et une mémoire est essentiellement un circuit séquentiel synchrone. Son cycle doit être inférieur au cycle de base du processeur afin de ne pas le ralentir.

Pour faire le lien avec le chapitre précédent nous admettrons que le temps d'exécution d'une micro-instruction est égal à un cycle de base. Pour simplifier, nous assimilerons cycle et micro-instruction.

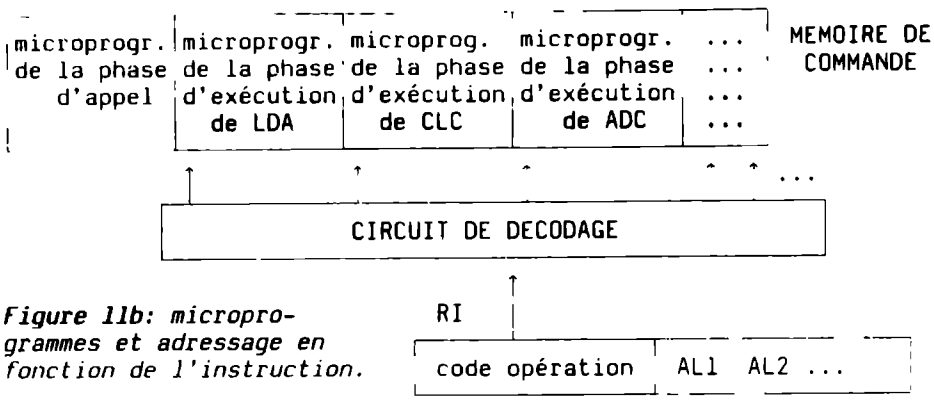


Figure 11b: microprogrammes et adressage en fonction de l'instruction.

11.4 – Structure de la micro-instruction.

La micro-instruction comprend trois parties principales:

- la partie **microcommandes** qui génère les microcommandes envoyées pendant le cycle concerné. Elle correspond aux fonctions du code opération et des champs AL de l'instruction,
- la partie **micro-instruction suivante** dont nous avons vu l'abandon dans le cas des instructions,
- la partie **retard**, sans équivalente dans les instructions (à part certains emplois de l'instruction NOP) qui elles se situent au niveau de l'architecture externe où le temps ne joue pas, du moins pas avec la même signification. Cette différence entre architectures externe et interne vaut d'être soulignée.

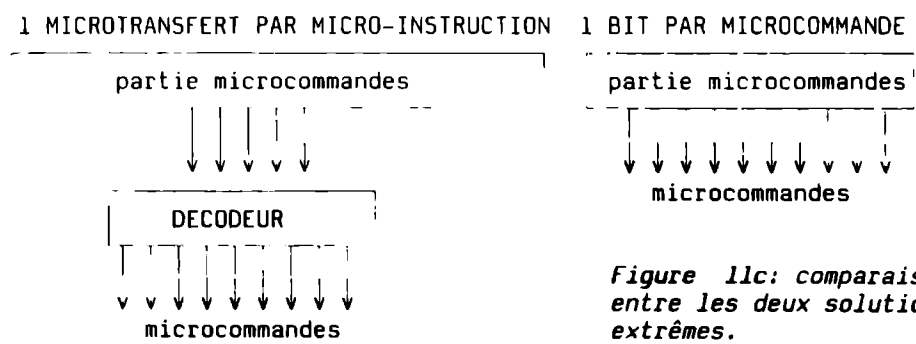


Figure 11c: comparaison entre les deux solutions extrêmes.

11.4.1 – Partie microcommandes.

L'ensemble des microcommandes à générer étant connu, se pose le problème de leur codage. Il existe deux solutions extrêmes:

- le codage avec **un bit par microcommande**. On aboutit ainsi à des micro-instructions très **longues** d'où le nom de microprogrammation horizontale (cas des 360/30) attaché à ce type de solution. Coûteuse, la solution offre cependant des performances potentielles excel-

lentes puisque toutes les microcommandes peuvent être envoyées simultanément;
- le codage avec **un microtransfert** ou un microtraitement par micro-instruction. Cette solution conduit à une micro-instruction courte, mais les performances sont minimales puisqu'il n'y a plus qu'une simultanéité réduite à sa plus simple expression. Elle nécessite de plus un circuit de décodage important (figure 11c), et si chaque micro-instruction est plus courte que dans le cas précédent, il en faut beaucoup plus dans la mémoire de commande.

La solution généralement retenue est bien évidemment une solution intermédiaire appelée **codage par champs**. L'ensemble des micro-instructions est partitionné en n groupes et chacun de ces n groupes dispose de son champ dans la micro-instruction. Deux microcommandes d'un même groupe ne peuvent être envoyées simultanément puisque dans chaque champ le code représente une et une seule microcommande du groupe. Par contre deux microcommandes de deux groupes différents pourront être envoyées par la même micro-instruction.

Le concepteur de l'architecture interne mettra donc dans un même champ des microcommandes qui par nature n'ont pas à être envoyées simultanément. C'est par exemple le cas des microcommandes ouvrant les portes de sortie de registres reliés par un bus: comme il ne peut y avoir qu'un seul registre source sur un bus, il n'y aura aucune baisse de performance due au codage par champs.

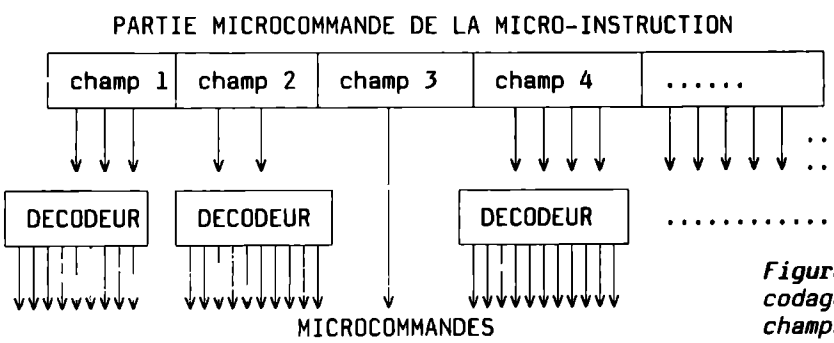


Figure 11d: codage par champs.

On utilise le terme de microprogrammation **verticale**, (cas du 360/25) "encoded", "packed", "soft", "softview", "functional" dans la littérature américaine pour ce type de microprogrammation. Les 64 de Bull disposaient de micro-instructions à cinq champs, les DPS7 ont sept champs.

On constate figure 11d qu'il faut un décodeur par champ, sauf cas particulier de champ dans lequel les micro-instructions seraient codées avec un seul bit par microcommande.

11.4.1.1 – Exemples.

Nous baserons nos exemples sur les machines MODELi, introduites dans le chapitre 9 et déjà utilisées dans le chapitre 10 pour le séquenceur câblé. La technique de codage adoptée est celle du codage par champs. Nous prendrons des champs correspondant autant que faire se peut à des microcommandes mutuellement exclusives, et sans chercher ici à optimiser au maximum.

Commençons par les bus qui sont par nature source de microcommandes mutuellement exclusives.

Le bus données BUS.D:

- reçoit des données des registres A, X, Y, S, et aussi de la mémoire centrale. Nous avons ainsi 5 microcommandes à coder, soit 6 configurations puisqu'il faut aussi représenter la situation où aucun registre ne débite sur le bus. Six configurations nécessitent un champ de 3 bits. Il reste deux codes pour prendre en compte d'autres sources, comme RINT2 dans MODEL4.

CODAGE	MICROCOMMANDE (champ CH1)
000	aucune
001	BUS.D ⟵ (MC), en fait déclenchée par la micro-commande qui positionne BUS.CL dans la machine MODEL1
010	BUS.D ⟵ (A)
011	BUS.D ⟵ (X)
100	BUS.D ⟵ (Y)
101	BUS.D ⟵ (S)
110	BUS.D ⟵ (RINT2) pour MODEL4

- envoie des données dans les registres A, RI, COg et COd (parties gauche et droite de CO), X, Y, S. Il envoie aussi des données sur une des entrées de l'UAL et dans la mémoire centrale. A ces neuf destinations il faut ajouter le cas où aucune microcommande n'est nécessaire, soit dix configurations différentes, donc 4 bits, six codes n'étant pas utilisés dans le cas de MODEL1.

CODAGE	MICROCOMMANDE	CODAGE	MICROCOMMANDE (CH2)
0000	aucune	0110	COg ⟵ BUS.D
0001	RM ⟵ BUS.D	0111	COd ⟵ BUS.D
	dans le cas de	1000	X ⟵ BUS.D
	MODEL3 ou MODEL4	1001	Y ⟵ BUS.D
0011	A ⟵ BUS.D	1010	S ⟵ BUS.D
0100	UAL ⟵ BUS.D	1011	RINT1g ⟵ BUS.D
0101	RI ⟵ BUS.D	1100	RINT1d ⟵ BUS.D

Dans le cas de MODEL1 il faut tenir compte des deux microcommandes qui servent à préciser si l'on veut lire ou écrire en mémoire centrale (commande des signaux BUS.CL et BUS.CE respectivement). D'où deux bits :

CODAGE	MICROCOMMANDES	CODAGE	MICROCOMMANDES (CH3)
00	aucune	10	BUS.CE haut (écrire)
01	BUS.CL haut (lire)		

Dans MODEL1 et MODEL2 le bus adresse BUS.A ne sert qu'à transférer le contenu de CO dans les circuits de décodage d'adresse de la mémoire centrale. Le champ associé dans la partie microcommande de la micro-instruction sera donc limité à un seul bit:

CODAGE	MICROCOMMANDES, MODEL1 et MODEL2 (champ CH4)
0	aucune
1	BUS.A ⟵ (CO) et MC ⟵ (BUS.A)

Si nous voulons intégrer MODEL3 et MODEL4, nous constatons qu'il faut un champ de deux bits:

00	aucun
01	BUS.A ⟵ (CO) et RAM ⟵ (BUS.A)
10	BUS.A ⟵ (RINT1) et RAM (BUS.A)

Autre composant à nécessiter une microcommande, le circuit d'incrémentation du compteur ordinal. Là encore un bit suffira:

CODAGE	MICROCOMMANDE (champ CH5)
0	incrémenter CO: CO ⟵ (CO)+1
1	aucune

Nous n'avons pas choisi le code zéro pour le cas où il n'y a aucune microcommande envoyée, afin de bien montrer que le codage est, au niveau où nous nous plaçons, purement arbitraire.

Les opérations que réalise l'UAL sont au nombre de 16 et donc codées sur 4 bits. Nous aurons:

CODAGE	OPERATION EFFECTUEE PAR L'UAL (CH6)
0000	aucune
0001	incrémentation de l'entrée venant du bus BUS.D
0010	additionner les entrées activées
0011	ET entre les entrées activées

Il nous reste les trois entrées de l'UAL: le registre accumulateur A, le bus données BUS.D et le report C. Nous prendrons a priori un bit par entrée potentielle pour un tel champ CH7:

zéro indiquera que l'entrée n'est pas prise en compte,
un indiquera que l'entrée est prise en compte: la microcommande correspondante est envoyée.

Le nombre de sorties de l'UAL varie selon ce que fait le circuit de décodage des instructions, et selon le MODELi considéré. Si le circuit de décodage se limite à déduire le code opération de l'UAL en fonction du code opération lu dans RI (figure 11f) il faudra pouvoir

coder trois sorties séparément (A, C, Z et N qui sont toujours positionnés ensemble). Si ce circuit de décodage ouvre automatiquement les portes d'entrée de IR en fonction du code opération de RI, il ne restera que la sortie A à coder. Dans le cas de MODEL4 il faut ajouter RINT2. Cet exemple montre que la frontière entre microprogrammé et câblé n'est pas rigide, mais constitue un compromis pour chaque architecture interne. On peut mettre plus ou moins de câblé et compléter par la microprogrammation. Ainsi [AME85] note que le HP 3000/48 contient deux fois plus de circuiterie (circuitry) que le 3000/37, mais requiert le même nombre de cycles d'horloge pour réaliser le même travail.

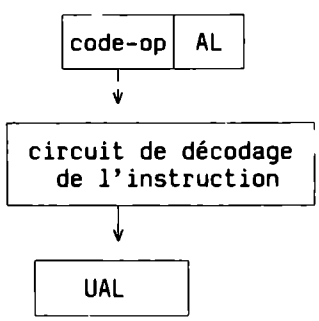


Figure 11f: seul le code opération de l'UAL est positionné.

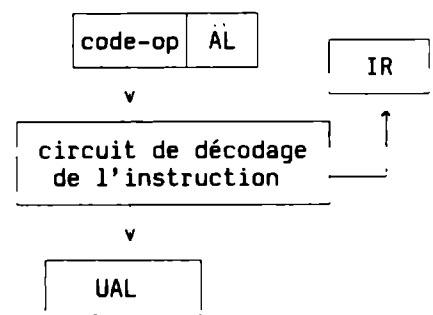


Figure 11g: le circuit de décodage positionne le code opération de l'UAL ainsi que les indicateurs.

Pour simplifier, nous supposerons que le circuit de décodage de l'instruction ouvre automatiquement les portes d'entrée du registre IR. Un seul bit suffira alors dans le cas de MODEL1, MODEL2 et MODEL3, deux seront nécessaires dans le cas de MODEL4:

CODAGE	MICROCOMMANDE (CH8)
0	aucune
1	A ⟵ (UAL)

CODAGE	MICROCOMMANDE (cas de MODEL4)
00	aucune
01	A ⟵ (UAL)
10	RINT2 ⟵ (UAL), seuls Z et N sont positionnés

Dans le cas de MODEL1 on aboutit à la partie microcommandes donnée dans le tableau 11h. Nous obtenons ainsi une partie microcommandes de 19 bits que nous illustrons par les exemples ci-après.

11.4.1.1.1 – Phase d'appel dans le cas de MODEL1.

En supposant que le microprogramme commence à l'adresse zéro en mémoire de commande, nous obtenons le résultat du tableau 11i. Le microprogramme se réduit à une seule microinstruction qui correspond au cycle [0,0] de la figure 10i dans le cas du séquenceur câblé.

CH1	CH2	CH3	CH4	CH5	CH6	CH7			CH8
BUS.D ← (x)	X ← (BUS.D)	MC L/E	MC ← (CO)	+1 CO	U A L opération	ENTREE A	UAL BUS.D	UAL C	SORTIE UAL
0 1 2	3 4 5 6	7 8	9	1 0	1 1 1 1 1 2 3 4	1 5	1 6	1 7	1 8

Tableau 11h: format de la partie microcommandes dans le cas de MODEL1.

11.4.1.2 – Codage de type instruction.

La micromachine peut faire l'objet d'une analogie plus poussée avec le processeur et disposera alors de micro-instructions structurées comme des instructions, avec un champ contenant le code opération, et des champs contenant les opérandes: numéros de registre, d'opérateurs, etc.

ADRESSE DE LA MICRO-INSTR.	PARTIE MICROCOMMANDES 0 1 2	3 4 5 6	7 8	9	1 0	1 1 1 1 1 2 3 4	1 1 1 1 5 6 7	1 8
0	0 0 1	0 1 0 1	0 1	1	0	0 0 0 0	0 0 0	0

Tableau 11i: phase d'appel dans MODEL1.

11.4.2 – Adresse de la micro-instruction suivante.

Le but de la partie adresse de la micro-instruction suivante est de définir la micro-instruction qui sera exécutée après la micro-instruction courante. Différents types d'adressage peuvent être implémentés selon les objectifs poursuivis: coûts, performances, simplicité, etc.

Cet adressage est généralement conditionnel, c'est-à-dire que la partie adresse comporte deux ou trois champs, selon que l'on fait ou non une hypothèse de séquentialité par défaut:

condition de saut	adresse en cas de condition réalisée

condition de saut	adresse si la condition est réalisée	non réalisée

L'adresse peut être absolue, mais pour les mêmes raisons que dans le

cas des modes d'adressage des instructions, une adresse relative est généralement préférée. Le HP 3000/37 permet par exemple de tester 32 conditions, chacun des champs "réalisée" et "non réalisée" contient deux bits qui définissent les registres contenant l'adresse de la micro-instruction suivante, le compteur ordinal des micro-instructions pouvant être ainsi référencé.

Pour illustrer cette partie adresse, prenons le cas de la phase d'appel de MODEL3 illustrée figures 10m et 10n dans le chapitre précédent. Nous choisirons une partie adresse sans hypothèse de séquentialité, en supposant que la mémoire de commande contient au plus 256 micro-instructions; d'où des adresses absolues de 8 bits. Les seules conditions de saut que nous ayons à prendre en compte pour l'instant sont:

- le passage de BUS.CM à l'état haut,
- le passage à la phase d'exécution de l'instruction, c'est-à-dire un branchement en fonction du contenu de RI.

Deux bits seront suffisants :

CODAGE	CONDITION	CODAGE	CONDITION
00	aucune	10	contenu de RI
01	BUS.CM haut		

La phase d'appel correspondante est donnée tableau 11j, la partie microcommandes passant à 20 bits du fait du champ CH4 permettant de remplir RAM à partir soit de CO, soit de RINT1 (bits 9 et 10).

ADRESSE de la MICRO-I	PARTIE MICROCOMMANDES 012 3456 78 90 1 2345 678 9	PARTIE ADRESSE 22 - 22222222 - 33333333 01 23456789 01234567
0	000 0000 00 01 1 0000 000 0	00 00000001 00000000
1	000 0000 01 00 1 0000 000 0	00 00000010 00000000
2	000 0000 00 00 1 0000 000 0	01 00000011 00000010
3	001 0101 00 00 0 0000 000 0	10 00000000 00000000

Tableau 11j: phase d'appel de MODEL3.

La première micro-instruction (adresse zéro) remplit le registre adresse de la mémoire centrale (RAM) et provoque un branchement inconditionnel (condition 00) à l'adresse 1 (contenue dans les bits 22 à 29).

La seconde micro-instruction envoie l'ordre de lecture à la mémoire centrale et provoque aussi un branchement inconditionnel à l'adresse 2, où on attend que BUS.CM passe à l'état haut (condition 01) pour passer à l'adresse 3. Si BUS.CM est bas en début de cycle, on ré-exécute cette micro-instruction d'adresse 2.

La micro-instruction à l'adresse 3 provoque le transfert de RM dans RI, l'incrémentation du compteur ordinal (bit 11) et un branche-

ment conditionné par le contenu de RI (condition 10).

Le circuit de décodage d'adresse de la mémoire de commande a alors comme entrées la partie adresse de la micro-instruction et la sortie du circuit de décodage de l'instruction dans RI (figure 11k). On notera, tableau 11j, qu'il y a peu de microcommandes activées simultanément. Une amélioration de la taille de la micro-instruction est certainement facilement réalisable.

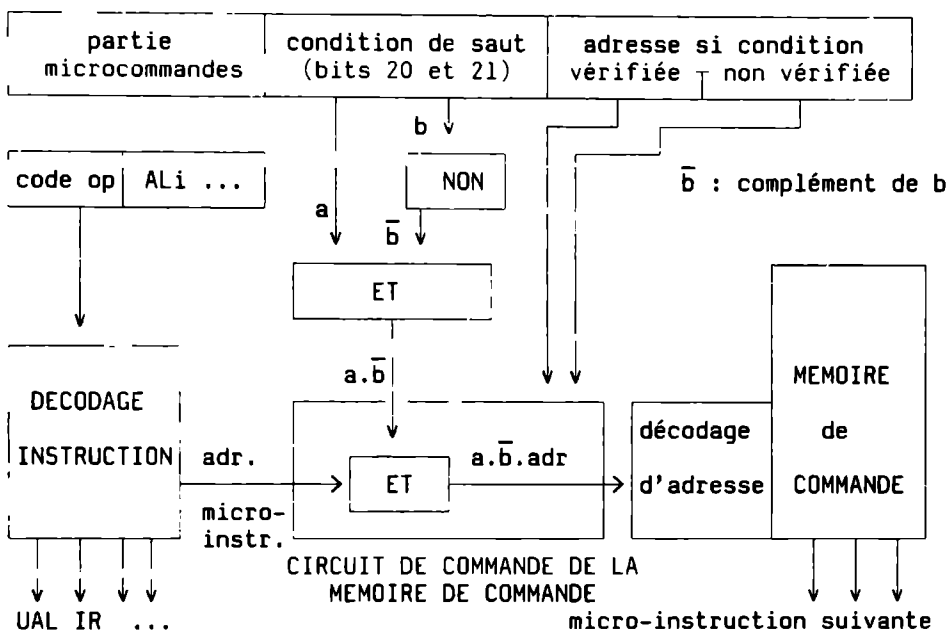


Figure 11k: circuits intervenant dans l'adressage de la micro-instruction suivante lorsque l'on passe à la phase d'exécution; a et b sont les deux bits de la condition de saut.

11.4.3 – Partie retard.

Le but de la partie retard est de définir l'instant où la micro-instruction suivante commencera à être exécutée. Ce retard peut être réalisé de façon :

- asynchrone: à chacun des bits de la partie retard est associé un laps de temps. La somme des différents laps de temps définit l'instant de démarrage de la micro-instruction suivante;
- synchrone: le retard est codé en nombre de cycles de base supplémentaires qu'il faut laisser passer avant de lancer la micro-instruction suivante. C'est ce cas de figure que nous illustrerons avec MODEL2.

MODEL2 a une mémoire centrale dont le temps d'accès Ta est égal à deux fois le cycle de base (cycle du processeur). Nous supposerons que son temps de cycle Tc est égal à quatre fois le cycle de base. La

phase d'appel et la phase d'exécution de AND# sont alors données dans les tableaux 11m et 11n, la partie retard codée sur deux bits permet des valeurs comprises entre zéro (aucun retard supplémentaire) et trois, ce qui suffit pour les retards liés au fonctionnement de la mémoire centrale.

MACHINE	TEMPS D'ACCES MC Ta	TEMPS DE CYCLE MC Tc	MC LIBRE EN DEBUT DE PHASE D'APPEL
MODEL1	<1	<1	oui
MODEL2	2	4	oui
MODEL3	inconnu	inconnu	oui
MODEL4	inconnu	inconnu	non

Tableau 111: caractéristiques des MODELi.

La première micro-instruction de la phase d'appel (adresse zéro dans le tableau 11m) comporte un retard de un cycle ce qui fait qu'entre le lancement de l'ordre de lecture et le transfert du contenu de RM dans RI il s'est bien écoulé deux cycles: celui pendant lequel la première micro-instruction s'est exécutée, plus celui d'attente créé par la partie retard.

ADRESSE de la MICRO-I	PARTIE MICROCOMMANDES 1 1111 111 1 012 3456 78 9 0 1234 567 8	PARTIE RETARD 12 90
0	000 0000 01 1 1 0000 000 0	01
1	001 0101 00 0 0 0000 000 0	00

Tableau 11m: phase d'appel dans MODEL2.

La micro-instruction à l'adresse x (tableau 11n) de la mémoire de commande est fonctionnellement vide car elle sert à attendre que la mémoire centrale soit libre: il faut en effet lire le second octet de AND#, mais depuis le dernier accès à la mémoire centrale il ne s'est écoulé que trois cycles, ceux provoqués par les micro-instructions aux adresses zéro et un. Comme le cycle de la mémoire est quatre fois plus grand que celui du processeur, il faut attendre encore un cycle de base avant de lancer la lecture suivante. On aurait pu coder un retard de un à l'adresse un en mémoire de commande, mais cela aurait ralenti les instructions qui n'ont pas à accéder une seconde fois à la mémoire centrale, comme TXA par exemple.

La micro-instruction à l'adresse x+1 comporte un retard de un pour la même raison qu'à l'adresse zéro. Le retard est aussi codé à un à l'adresse x+2 car la micro-instruction qui suit est celle d'adresse zéro, et elle suppose qu'au début de son exécution la mémoire centrale soit libre. En x+1 on envoie une requête à la mémoire centrale, et l'exécution des micro-instructions d'adresse x+1 et x+2 requiert bien quatre cycles de base, ce qui assure la disponibilité de la mémoire centrale lorsqu'on revient en phase d'appel pour l'instruction suivante.

ADRESSE de la MICRO-I	PARTIE MICROCOMMANDES 1 1111 111 1 012 3456 78 9 0 1234 567 8	PARTIE RETARD 12 90
x	000 0000 00 0 1 0000 000 0	00
x+1	000 0000 01 1 0 0000 000 0	01
x+2	001 0100 00 0 0 0011 110 1	01

Tableau 11n: phase d'exécution de AND# avec MODEL2.

11.5 – Historique.

Le créateur de la microprogrammation est Wilkes qui publia un article célèbre en 1951. Selon Wilkes, une instruction en langage machine pouvait être découpée en une série d'étapes élémentaires appelées "opérations". Les machines microprogrammées furent commercialisées à partir de 1964 avec la série 360 d'IBM.

11.6 – Microprogrammation à deux niveaux.

La taille des micro-instructions tend à augmenter lorsque le processeur est puissant et la microprogrammation horizontale. Pour diminuer la taille globale de la mémoire de commande on crée deux niveaux de mémoires de commande.

Dans le niveau le plus bas on mettra la partie longue des micro-instructions, celle qui génère les microcommandes et se retrouve dupliquée dans la microprogrammation à un seul niveau. On les appellera des nano-instructions pour traduire le changement d'échelle. D'où le terme de nanocode pour nommer l'ensemble des nano-instructions.

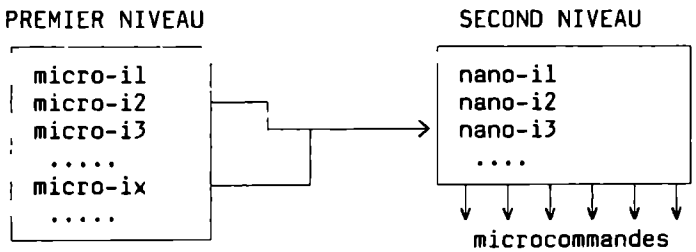


Figure 11p: mémoire de commande à deux niveaux.

Les micro-instructions sont plus nombreuses mais plus courtes. Le 68000 de Motorola, le QM-1 de Nanodata, le Mitra 15 de l'ex-CII sont des exemples de processeurs avec des mémoires de commande à deux niveaux.

Comme l'indique le tableau 11q les mémoires de commande peuvent comprendre une partie en mémoire vive (1 Ko sur VAX-11/780 par exemple) dans laquelle soit on charge la majeure partie du microcode lorsque l'on démarre l'ordinateur, soit on charge les microprogrammes non résidents dont on a besoin à un moment donné.

PROCESSEUR	NOMBRE DE NIVEAUX	TAILLE DU MOT en bits	TAILLE DE LA MEMOIRE MEM(MEV) en Kmots
VAX-11/780		96 + 3	4 (1)
VAX-11/750		80	6
VAX-11/730		24	16 (1)
68000	2	10/70	
8086	1	21	
370/158		72	8 Ko
HP 9000		38	9
4341-1 ou 2		40	128
DPS7		56 + 8	12 à 64
3031		72	8
B 1776		16	
Nanodata QM1	2	18/360	16 (1)
360/25		16	8
360/30		60	
360/85		108	2-5

niv1/niv2
xx + pp : pp bits de parité

Tableau 11q: exemples de tailles de micro-instructions.

11.7 – Microprocesseurs et microprogrammation.

Les termes de microprogrammation et microprocesseur étant proches l'un de l'autre, le lecteur pourrait penser que lorsque l'on écrit des programmes en langage machine sur des microprocesseurs, on fait de la microprogrammation. Il n'en est rien bien sûr puisque le langage accessible au niveau de l'architecture externe n'est pas celui que constituent les micro-instructions, qui elles se trouvent à l'intérieur du matériel, au niveau de l'architecture interne, comme le représente la figure 11r.

L'ensemble des microprogrammes d'un ordinateur est appelé **micro-code ou élasticiel (microcode, firmware en américain).**

Il y a d'ailleurs des microprocesseurs câblés (les Z8000 et 6502 par exemple), et des microprocesseurs microprogrammés comme le Z-80, le 68000, le NS16032. Utilisé dans les PC XT/370, le 68000 y est microprogrammé non plus pour exécuter son jeu d'instructions natif, mais pour exécuter les instructions de base du jeu du 370 d'IBM.

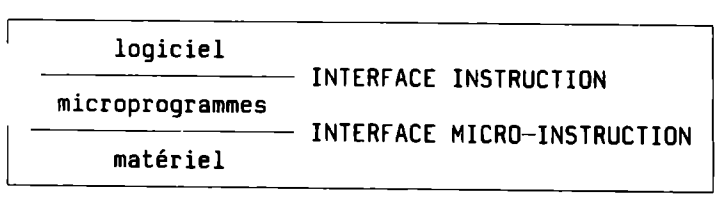


Figure 11r: instructions et micro-instructions se trouvent à deux niveaux différents d'architecture.

11.8 – Microprocesseurs en tranche.

Si l'on utilise en parallèle plusieurs microprocesseurs de n bits, on obtient un multiprocesseur ou un multisystème (cf. chapitre 17). Pour réaliser un processeur en utilisant des modules mis en parallèle, on prend des microprocesseurs en tranche. La 1100/60 de Sperry a ainsi un processeur 36 bits composé de 9 de ces microprocesseurs, chacun travaillant sur 4 bits. Les microprocesseurs en tranche ne sont donc pas réellement des microprocesseurs car une tranche isolée ne peut rien faire à elle seule. Ils permettent de réaliser des processeurs plus rapides et/ou spécialisés que les microprocesseurs.

11.9 – Comparaison entre câblé et microprogrammé.

Un même algorithme peut être soit programmé, soit microprogrammé, soit câblé: la solution câblée offrira la performance maximale du point de vue de la vitesse. Sur le plan de la souplesse, la microprogrammation se révèlera par contre supérieure: il est plus facile de modifier le contenu d'une mémoire, qu'elle soit vive ou morte, que de refaire un circuit.

La microprogrammation a jusqu'ici été présentée comme une alternative au câblage. Si ceci était totalement vrai au début, il n'en est plus ainsi aujourd'hui, la microprogrammation ayant dépassé le niveau de l'interface instruction (figure 11s), niveau fonctionnel que le câblé ne dépasse jamais, sauf cas très particulier. C'est ainsi que les fonctions suivantes sont microprogrammées:

- "assist" pour le tri DFSORT Release 7 ([WAG85]) sur les IBM 308X,
- des interpréteurs comme l'APL sur 370/148, le Basic sur Réalité 2000 d'Intertechnique,
- le distributeur (cf. Tome II) sur les PRIME, DPS7,
- le traitement de premier niveau des interruptions sur DPS7, sur XA d'IBM (on passera ainsi selon [WAG85] de 50 à 1 instruction, c'est-à-dire de 300 à 70 cycles),

IBM joue beaucoup avec ce microcode pour gêner les fabricants de compatible.

Cette évolution va jusqu'à la possibilité de microprogrammer offerte aux utilisateurs, en particulier dans les minis du haut de gamme.

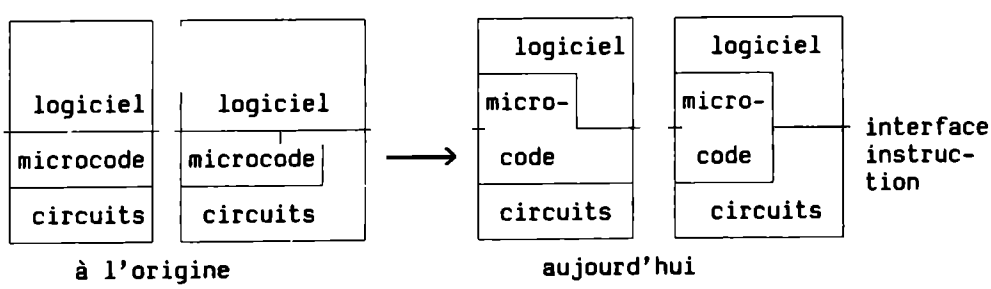


Figure 11s: évolution de la microprogrammation.

chapitre 12

compatibilité

Le terme de compatibilité n'est pas nouveau dans cet ouvrage, nous l'avons souvent utilisé au cours des chapitres concernant l'architecture externe. Il est cependant utilisé dans des contextes et à des niveaux très différents.

12.1 – Définition de base.

Dire qu'un ordinateur A est compatible avec un ordinateur B signifie qu'il a la même architecture externe. Ils peuvent en effet:
- supporter les mêmes logiciels,
- disposer des mêmes sous-systèmes périphériques puisque ces derniers respectent les mécanismes et les interfaces d'entrée/sortie définis du point de vue du sous-système central. Nous les étudierons dans le Tome III.

12.2 – Compatibilité dans une gamme.

Les différents modèles d'une même gamme sont compatibles selon la définition précédente. Cette compatibilité peut cependant ne pas être totale. On parlera entre deux modèles de:
- compatibilité **ascendante** (upward) lorsque les modèles plus puissants sont compatibles avec les modèles de bas de gamme, mais que l'inverse n'est pas vrai. C'est le cas de la gamme INTEL avec les 8086, les iAPX286 et les iAPX386. On parlera aussi au niveau de l'architecture interne de compatibilité croissante **sur le site** (on site upgradable) lorsque le changement de modèle n'oblige pas à faire sortir physiquement l'ancienne machine de la salle ordinateur pour faire entrer la nouvelle, mais conduit simplement à réaliser une mise à niveau (upgrade) de l'ancien modèle;
- compatibilité **descendante** (downward) lorsqu'un modèle moins puissant a la même architecture externe qu'un modèle plus puissant. La compatibilité est alors généralement valable dans les deux sens, mais pas obligatoirement.

12.3 – Compatibles IBM 370 et gammes suivantes.

Il est courant de remplacer un sous-système central IBM par un compatible en gardant la même périphérie. Il y a en 1985 plus d'une dizaine de sociétés commercialisant des compatibles IBM, les constructeurs dominants dans le haut de gamme étant des japonais (Hitachi et Fujitsu). Amdahl, Siemens, ICL, commercialisent des compatibles que leur fournit Fujitsu, le cas d'Amdahl étant légèrement différent puisqu'il participe à la conception. BASF, NAS, Olivetti commercialisent des machines fabriquées par Hitachi.

Au niveau des machines moyennes compatibles avec les 370 et leurs évolutions, on trouve comme concepteurs IPL, Four Phase, ELBIT, Formation; ces matériels sont revendus par eux-mêmes et par Olivetti, Cambex, CSS, NIXDORF, etc.

Ouvert, lorsque le matériel et le logiciel n'étaient pas facturés séparément, ce qui avantageait les fabricants du seul matériel puisqu'ils n'investissaient pas dans le développement de logiciels, ce marché des compatibles connaît actuellement des difficultés pour nombre de fabricants ou vendeurs: Magnuson a disparu, CDC a cessé de vendre ses Omega, ITEL a été repris par NAS, STC a renoncé en 1984 alors qu'elle était sur le point de sortir ses premiers compatibles, Trilogy a aussi renoncé en juin 1984.

12.4 – Rôle du logiciel.

Le logiciel et en particulier les systèmes d'exploitation (logiciels généralement fournis par le constructeur afin d'offrir un niveau de service beaucoup plus élevé que celui offert par les simples instructions machine que nous venons de voir) viennent compliquer la situation simple exposée dans la définition de base.

Cette définition de base intéresse en effet les responsables qui ont la charge d'acquérir les ordinateurs. Les compatibles offrant un meilleur rapport performance/prix sont pour eux un moyen de mettre le constructeur dominant en concurrence, en posant le problème au niveau du matériel uniquement.

Pour l'utilisateur normal, c'est-à-dire celui qui se préoccupe de savoir si son programme d'application fonctionnera sans problème en passant d'une machine A à une machine B, le problème est plus compliqué car il lui faut tenir compte des niveaux d'architecture qu'imposent les systèmes d'exploitation (cf. les niveaux présentés par [INT81] dans le chapitre 1).

Pour que son programme soit **transportable**, il faut que l'interface offerte par le nouveau système soit compatible avec celle du précédent. Or ceci peut aussi être réalisé en passant sur une nouvelle machine ayant une architecture externe différente: c'est le cas du système d'exploitation UNIX qui tourne aussi bien sur des 68000, des VAX, des Perkin-Elmer, des PRIME, etc. Encore convient-il de se méfier des différentes versions d'UNIX.

Mais le cas contraire existe aussi et est illustré par la gamme 370 d'IBM: il existe plusieurs systèmes d'exploitation sur cette gamme et ses dérivées, principalement DOS et MVS. Ces deux systèmes d'exploitation ne sont pas compatibles et l'on ne peut pas y transporter

des programmes exécutables bien qu'on ait affaire au même jeu d'instructions. Même au niveau source il y a des différences, et le langage de commande (JCL) n'est pas le même. Passer de l'un à l'autre constitue une **conversion** non négligeable.

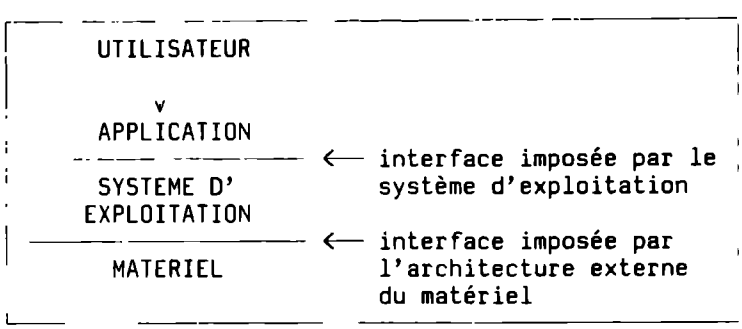


Figure 12a: compatibilité au niveau application.

En faisant intervenir les systèmes d'exploitation dans cette discussion de la compatibilité, nous voyons que le problème se pose à différents niveaux, et l'on entendra parler de compatibilité:
- au niveau **exécutable**, c'est-à-dire au niveau des programmes générés par les éditeurs de lien (cf Tome V);
- au niveau **objet**, c'est-à-dire au niveau des programmes générés par les compilateurs. Si cette compatibilité n'est pas assurée, mais que par contre on dispose de la compatibilité au niveau source, il faudra recompiler les programmes sur la nouvelle machine;
- au niveau **source** si les compilateurs acceptent la même écriture des programmes;
- au niveau **fichier** si les organisations de données sont identiques;
- au niveau **langage de commande**.

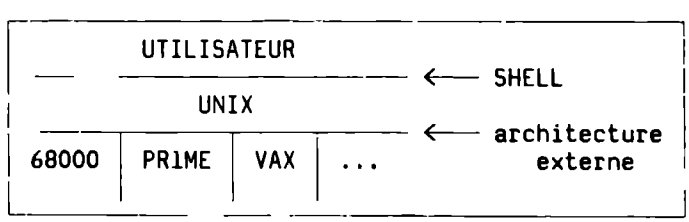


Figure 12b: compatibilité source grâce au système d'exploitation.

12.5 – Rôle des périphériques.

Si nous continuons avec notre utilisateur qui se préoccupait du transport de ses applications, il devra se soucier de la compatibilité au niveau des supports qui lui permettront de porter physiquement ses programmes et ses données sur la nouvelle machine.

Sur les gros ordinateurs, la solution est généralement offerte par les bandes magnétiques et de plus en plus par les télécommunications.

Au niveau des micro-ordinateurs on ne dispose pas de bande magnétique et les communications ne sont pas toujours possibles. Le support

roi est la disquette, mais les formats et les caractéristiques les rendent la plupart du temps incompatibles si l'on n'y prend pas garde. Ainsi entre un Apricot sous MS-DOS (le système d'exploitation des PC) et un PC sous le même MS-DOS on aura le problème (en 1985) des disquettes 3 pouces et 5 pouces.

Si l'on considère, toujours dans le domaine de MS-DOS, un Sirius de Victor et un PC avec tous deux des disquettes 5 pouces, on aura à nouveau des problèmes car sur le PC elles tournent à vitesse constante quelle que soit la piste, alors que sur le Victor la vitesse change selon les pistes (le Tome IV apportera des explications plus détaillées). Résultat: le transport des disquettes ne sera pas possible.

Si l'on fait intervenir, dans ces PC, les problèmes de clavier et d'écran, la discussion sur la compatibilité devient encore plus complexe. Voir [MAR85], [CHA85], [VER85] et [DID85].

En conclusion il convient de bien poser le problème lorsque l'on parle de compatibilité. Une réponse affirmative peut ne correspondre que partiellement à la réalité pratique. Il y a toujours un "certain" niveau de compatibilité.

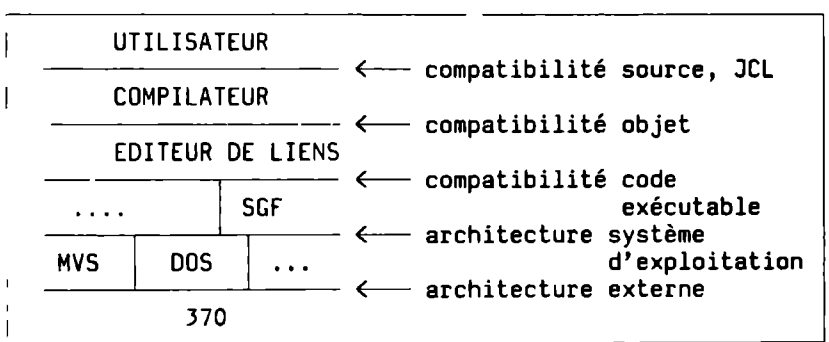


Figure 12c: incompatibilité introduite par l'architecture du système d'exploitation alors qu'il y a compatibilité au niveau de l'architecture externe du matériel.

12.6 – Sous-systèmes périphériques compatibles.

Bien que ce tome ne porte pas sur les sous-systèmes périphériques, il nous semble intéressant de compléter ce chapitre en indiquant que, tout comme il y a des compatibles pour les sous-systèmes centraux, il existe des fabricants de compatibles pour les sous-systèmes périphériques. Ce ne sont d'ailleurs pas les mêmes. Ainsi dans le domaine des disques sur les 370 on trouve Memorex, CDC, STC, etc.

chapitre 13

performances

Le problème des performances, donc de la vitesse de traitement, est un problème très complexe car il fait intervenir de nombreux éléments; il n'est pas représentable par un nombre unique, alors que les utilisateurs voudraient un tel nombre unique, facile à comprendre et .... vrai.

13.1 – Les éléments influençant les performances.

Les principaux facteurs qui ont une incidence sur les performances intrinsèques du matériel sont répertoriés ci-dessous. Il convient effectivement de bien séparer les performances du matériel de celles observables par les utilisateurs: pour eux, au matériel viennent s'ajouter les effets du système d'exploitation, des compilateurs, des méthodes d'accès, des librairies mathématiques ou autres (run time libraries en américain), etc. Le lecteur trouvera dans [FRI85] une étude très accessible concernant les 8086/8088/8087. Nous mentionnerons ici les facteurs suivants:

- le **cycle de base** du processeur, lorsqu'il est synchrone, joue un rôle important puisque tout se mesurera en nombre de cycles. Il faut cependant savoir ce qui se passe à l'intérieur d'un cycle, car d'une architecture interne à l'autre la quantité de traitement par cycle peut être très différente. Il en est de même avec les moteurs automobiles dont la vitesse de rotation ne donne pas systématiquement la puissance: un moteur de grosse cylindrée tournant à 4000 tpm (tours par minute) sera plus puissant que le moteur d'une petite cylindrée qui tourne à 7000 tpm.
  Dans l'annexe C on constate ainsi que le 3033S d'IBM et le 470V/7C, un compatible, sont estimés avoir la même puissance relative (132) alors que le premier a un cycle de base de 57 ns contre 29 ns pour le second. Si l'on appliquait le rapport des cycles, le 470 devrait être deux fois plus puissant.
  Si l'on compare maintenant le 3033S avec son cycle de 57 ns et le 3083E qui a un cycle de 26 ns, le 3033S est estimé plus puissant, malgré son cycle plus lent;
- le **cycle majeur**, cycle de la mémoire centrale joue aussi un rôle important que nous soulignerons au chapitre 18. Si la mémoire est

lente le processeur en sera ralenti puisqu'il est forcé d'aller y rechercher les instructions et les données à traiter;
- le **jeu d'instructions** a un rôle essentiel avec les **modes d'adressage**;
- le nombre de **registres** et la façon dont ils sont implémentés;
- la proportion de **microprogrammation** utilisée pour interpréter les instructions. Les machines haut de gamme ont un pourcentage de microprogrammation plus faible;
- la façon dont les **microprogrammes** sont **implémentés**. Ils peuvent résider dans une mémoire rapide spéciale (la mémoire de commande que nous avons vue) ou dans la mémoire centrale (cas du 370/135) qui joue alors le rôle de mémoire de commande;
- un autre élément concerne le nombre et la structure des **opérateurs**, des coprocesseurs numériques. Nous rappelerons les options "flottant câblé" vues précédemment et dont l'effet est estimé dans le tableau sur les performances en Whetstones (tableau 13g);
- la **largeur du chemin de données** joue un rôle qu'il faut rapprocher de celui du cycle de la mémoire centrale. Les gammes de Burroughs et d'IBM utilisent fréquemment cet élément pour réaliser des modèles de puissances différentes. La gamme 370 qui ne comporte pas moins de treize modèles a des transferts entre mémoire et processeur qui vont de 2 à 16 octets selon les modèles. Les données sont transférées sur 16 bits dans le 8086, sur 8 bits dans le 8088.
  Le **multiplexage** des bus de données et d'adresse a un effet analogue: le 6502 a deux bus séparés, le 8086 se vend en deux versions, l'une avec adresse et données multiplexées sur les mêmes fils (version "basse"), l'autre sans multiplexage (version "haute");
- tous les mécanismes d'**anticipation** (pipelining) et de **parallélisme** que nous étudierons dans la suite de ce tome.

13.2 – Unités de puissance.

Le titre de cette section montre qu'il n'y a pas d'unité de puissance **unique**. Nous verrons successivement les MIPS, les Gibson mix, les puissances relatives, les FLOPS, les jeux d'essai.

13.2.1 – Les MIPS.

L'unité dont on parle le plus est le KIPS (kilo instruction par seconde) ou MIPS (million d'instructions par seconde). Le gros problème posé par ces MIPS est que l'on ne sait pas toujours quelles sont les instructions exécutées lors de la mesure. Il est bien évident que l'on n'obtiendra pas le même résultat sur un IBM 43XX en exécutant des MVCL qui déplacent un million d'octets par instruction qu'en exécutant des LR qui transfèrent le contenu d'un registre dans un autre registre. De même avec le 6502, il y aura par seconde moins de DECAX que de LDA# exécutées: DECAX nécessite 7 cycles alors que LDA# n'en prend que deux. [MDO84] note qu'il n'est pas rare qu'un compilateur COBOL exécute ses instructions avec un débit double de celui obtenu par les programmes qu'il génère.

Pour qu'une puissance en MIPS signifie quelque chose de précis, il faut connaître le sous-ensemble du jeu d'instructions utilisé. Des

comparaisons entre des processeurs de même architecture externe pourront être exprimées en MIPS si le sous-ensemble du jeu d'instructions utilisé est précisé.

Quand on compare des processeurs d'architectures externes différentes, la comparaison en MIPS relève du grand art. Les MIPS donnent dans ce cas un ordre de grandeur. Dans le Monde Informatique du 25 octobre 1982 on lit page 2 "... offrant 3 MIPS de puissance (2,2 MIPS IBM) pour ..." à propos d'un nouvel ordinateur de Hewlett Packard, le 1000/A900. On voit dans cet exemple que les MIPS n'ont pas la même signification selon les architectures externes et certains ont le mérite de l'indiquer.

Un autre exemple de la valeur relative de ces MIPS est la polémique qu'il y a eu en 1982 entre le constructeur CDC et T. Henkel de la revue Computerworld. T. Henkel publie chaque année des puissances en MIPS (annexe C). Le rapport d'appréciation entre CDC et le journaliste (cf. tableau 13a) varie dans un rapport de moins de 1 à plus de 5 selon le modèle considéré.

ORDINATEUR	MIPS selon CDC	MIPS selon Computerworld
170/720	1,4	0,79
170/730	2,3	1,1
170/740	4,2	2,7
170/750	7,5	5,7
170/760	10,0	7,8
170/825	1,5	0,79
170/835	3,5	1,2
170/855	8,0	1,5
170/865	11,0	2,3
170/875	19	2,4
176	15	17,7

Tableau 13a: puissances en MIPS selon le constructeur et selon un journaliste.

Le lecteur aura pris conscience de la relativité des puissances exprimées en MIPS, surtout lorsque l'on compare des architectures différentes. Il faut aussi avoir en tête le fait que les MIPS ne traduisent que la puissance du sous-système central. Ils ne prennent pas en compte les sous-systèmes périphériques, les logiciels. La réalisation, sous forme microprogrammée, de certaines fonctions du système d'exploitation (cas des 370 et successeurs) fausse encore plus ces comparaisons basées sur les MIPS.

Les MIPS sont aussi utilisés pour exprimer la tendance dans l'évolution des coûts (tableau 13b). Des analystes du marché expriment la taille du parc installé en MIPS (tableau 13c), et [KLE85] note que "le nombre de MIPS installés dans les PC (ordinateurs personnels) est un ordre de magnitude supérieur à celui des gros/moyens ordinateurs". De grosses sociétés comme EDF parleront de leur capacité installée en MIPS: 669 MIPS dans le Monde Informatique du 16 septembre 1985.

13.2.2 – Les Gibson mix.

Plus anciens que les MIPS et tombés en désuétude, les Gibson mix (tableau 13e) définissent un mélange pondéré d'instructions dont le temps d'exécution traduit la puissance du processeur considéré.

Si l'on considérait le 6502, on aurait des difficultés à y adopter le Gibson mix puisqu'il n'y a pas de flottant. Lorsque l'on lit 33 % d'additions entières, prendrons-nous ADC# ou ADC*X ? ADC# nécessite 2 cycles, ADC*X en prend 6. Le Gibson mix n'est pas sans ambiguïté lui non plus.

ANNEE	SYSTEME	MIPS	$ PAR MIPS
1970	370/165	1,8	2 475 000
1976	370/168	2,5	2 240 000
1982	3081K	14,0	315 000
1984	3083	4,4	275 000
1984	43XX		150 000
1985			170 000
1989			70 000

SYSTEME	PRIX 1985
VAX 8600	83 000
VAX-11/780	145 000
HARRIS 1200	42 000
MV/10000	70 000
MV/10000SX	62 000
MV/20000-1	42 000
MV/20000-2	33 000

Tableau 13b: prix du MIPS en fonction du temps et du type d'ordinateur.

ANNEE	PUISSANCE EN MIPS
1976	1 884
1977	2 685
1978	4 230
1979	7 507
1980	11 034
1981	15 052
1982	19 789

Tableau 13c: puissance des machines de type 370 installées, en MIPS selon IDC.

13.2.3 – Puissances relatives.

Les MIPS étant ambigus, du fait de leur signification apparemment claire (un **nombre** d'instructions par seconde) et de ce que cache réellement le terme **d'instruction**, des puissances relatives ont été utilisées entre deux modèles de même architecture. Vers 1978 IBM a commencé à donner des rapports entre ses différents modèles, en particulier pour indiquer la puissance des nouveaux par rapport aux anciens. Bull le faisait déjà depuis plusieurs années pour les DPS8 et leurs prédécesseurs. En 1981, avec l'annonce des 4321 et 4331-1X, IBM a donné deux séries de rapports, l'une valable dans le contexte de la gestion, l'autre dans le domaine scientifique, et on observera que ces rapports évoluent différemment dans le tableau 13d. L'utilisateur est ainsi sensibilisé au fait qu'il n'y a pas de chiffre unique.

Pendant des années Bull a publié un seul chiffre de puissance relative pour ses DPS8, alors que l'opérateur commercial était le même

du haut en bas de la gamme. D'où l'incompréhension des utilisateurs qui essayaient de vérifier les chiffres publiés par le constructeur.

MODELE	4321	4331-1	4331-11	4331-2	4341-10	4341-1
RAPPORT						
EN GESTION	1	1	1,5	2	3,2	4
EN SCIENTIFIQUE	0,7	0,7	1,4	1,4		

Tableau 13d: puissances relatives fournies par IBM.

GIBSON SCIENTIFIQUE		GIBSON COMMERCIAL [ROG76]	
POIDS	INSTRUCTION	POIDS	INSTRUCTION
33	addition binaire entière	25	déplacement (MOVE)
0,6	multiplication bin. ent.	4	édition
0,2	division binaire entière	8,2	addition /soustraction
7,3	addition flottante	7	lancement d'entrée/sortie
4	multiplication flot.	0,6	multiplication
1,6	division flottante	0,2	division
6,5	branchement condit.	31	branchement conditionnel
4	comparaison	24	comparaison
17,5	chargement de registre		
4,6	décalage simple		
1,7	instruction logique		
19	incrémentation registre		

Figure 13e: composition des Gibson mix en % de types d'instructions.

13.2.4 - KOPS et MFLOPS.

Les KOPS (kilo opération par seconde) et MFLOPS (millions d'opérations en flottant par seconde) n'expriment plus un nombre d'instructions. Utilisés dans le domaine scientifique, ils expriment ce que l'utilisateur attend de ces ordinateurs, des résultats en flottant. Certaines instructions ne produisent aucun résultat, les instructions de transfert par exemple. D'autres en produisent plusieurs, les instructions vectorielles. Le tableau 13f donne quelques puissances en MFLOPS. Les plages de puissance, la différence entre puissance théorique et puissance réellement observée s'expliquent par:

- le fait que tous les calculs ne sont pas capables d'utiliser les ressources de ces ordinateurs ou opérateurs vectoriels à cent pour cent. Nous verrons dans le chapitre sur l'anticipation qu'il faut du temps pour qu'un pipeline atteigne sa vitesse de croisière;
- dans les processeurs vectoriels, les instructions scalaires ralentissent beaucoup le traitement: 1 % d'instructions scalaires dans un programme peut représenter un temps d'exécution dix fois plus important que celui passé dans la partie vectorielle de ce même programme (cf. [KLE85]). Il faut donc rendre possible l'exécution simultanée des instructions scalaires et vectorielles. C'est par exemple ce que permet de faire le 205 de CDC, mais encore faut-il que le programme s'y prête.

MODELE	PUISSANCE THEORIQUE	PUISSANCE RELLE	PRIX 85 en M$	(1)
CRAY 2	1 600		17,6	12 k$
CRAY XMP	1 400		5 à 14	30 k$
AMDAHL 1400	1 140			20 k$
SX-2 (NEC)	1 300			15 k$
CRAY-1	80 - 140	235		
CYBER 205	50 - 200		6 à 15	35 k$
STAR 100	25 - 150	168		
DENELCOR 1500	10 - 160		1 à 5	
NAS 9100	50		2 à 5	
ILLIAC IV	40 - 80	20		
AP-120B	6 - 12	ASSM: 59		
		FORT: 10		
CDC 7600	5 - 15	33		
IBM 370/168	2 - 4	0,75		
VAX-11/780	0,5			
PDP-11/70	0,2	0,09		

Tableau 13f: exemples de puissances en MFLOPS.

(1) par MFLOPS

13.2.5 – Whetstone, US Steel, etc.

Une autre façon d'appréhender la puissance des sous-systèmes centraux est d'exécuter des programmes de référence, des bancs d'essai (benchmark). Quelques programmes sont ainsi devenus des références très utilisées.

Dans le domaine scientifique on utilise beaucoup le programme de M. Whetstone. Ecrit en Fortran il prend donc aussi en compte les performances du compilateur et de quelques librairies mathématiques. Le tableau 13g donne les puissances Whetstone, en nombre de milliers d'instructions du programme exécutées par seconde. [INT85A] contient ce programme.

On constate les effets de l'optimisation du code généré par les compilateurs sur l'exemple du 32/8780 de GOULD-SEL: la puissance apparente est presque triplée. L'effet des options flottantes câblées: 63 % sur le VAX-11/780, 50 pour la MV/4000. La baisse de puissance en double précision peut être très importante, 35 % avec l'option flottant câblé, 67 sans cette option sur le VAX-11/780.

Les chiffres de ce tableau peuvent être discutés dans la mesure où nous ne les avons pas produits nous-mêmes. Ils proviennent d'articles différents ([BUT83], etc.) et doivent être considérés avec précaution.

La façon de passer ces programmes peut aussi donner lieu à optimisation et donc contestation. Dans de nombreuses machines mieux vaut les passer seuls (en monoprogrammation) pour obtenir un bon résultat. Des paramètres système (tranche de temps par exemple) peuvent aussi être définis pour le besoin et ne pas correspondre au cas normal de fonctionnement.

Si le Whetstone est très populaire dans le domaine scientifique, l'US Steel benchmark est souvent utilisé dans le domaine de la gestion, mais sans atteindre l'importance du Whetstone.

CONSTRUCTEUR	MODELE	NOMBRE DE BITS	PUISSANCE EN WHETSTONES SIMPLE PRECISION	DOUBLE PRECISION
GOULD-SEL	32/8780	32	6 659	
GOULD-SEL	32/8780 *	32	17 477	
NORSK DATA	ND540	32	4 100	
PERKIN ELMER	3250	32	3 000	
DATA GENERAL	MV/10000	32	2 500	
NORSK DATA	ND520	32	2 100	
PRIME	850	32	1 650	
IBM	4341-II	32	1 500	
RIDGE	32	32	1 500	
NORSK DATA	ND500	32	1 400	1 160
DATA GENERAL	MV/8000 V	32	1 150	875
DEC	VAX11/780 V	32	1 140	730
PERKIN ELMER	3240 V	32	1 100	750
DEC	VAX11/780	32	700	230
NS	32032 10 Mh	32	670	
DATA GENERAL	MV/4000 V	32	600	400
BULL	6/92 V	32	500	
DEC	VAX11/750	32	400	140
DATA GENERAL	MV/4000	32	400	200
DEC	PDP11/44 V	16	314	231
MOTOROLA	68000 8 Mhz	16/32	70	

Tableau 13g: puissances en kilo Whetstones.

V : avec option flottante câblée
* : avec code optimisé

Dans le domaine des micro-ordinateurs, chaque revue ou presque, des boutiques, publient des résultats basés sur des programmes synthétiques qu'ils ont eux-mêmes définis (cf. Minis et Micros numéro 210, page 65, par exemple). L'approche relève du même principe que le Whetstone.

Signalons par exemple les chiffres publiés par la revue "Byte", chiffres correspondant au temps d'exécution, en secondes, d'un algorithme de recherche de nombres premiers dans un tableau :

- 0,52 en assembleur sur iAPX286,
- 1,12 en assembleur sur 68000,
- 1,14 en C sur iAPX286,
- 1,40 en C sur VAX-11/780.

D'autres programmes, en Pascal, appelés "Berkeley Architecture Benchmark", ont donné l'iAPX286 comme étant deux fois plus rapide que le 68000 et 1,5 fois plus rapide que le VAX-11/780.

13.2.6 – Autres unités de puissance.

Dans le cadre des ordinateurs de cinquième génération on évalue les performances en terme de LIPS (Logical Inferences Per Second) c'est-à-dire de nombre de déductions que ces machines réalisent par seconde. Une opération de déduction correspond à un nombre d'instructions classiques compris entre 100 et 1000 (cf [KUR82]).

Moins récent que les LIPS, la "physique" du logiciel (software physics) de Kolence ne semble pas sortir d'un cercle restreint d'initiés. Le lecteur trouvera une étude intéressante, faite par un constructeur, dans [MD084].

13.3 – Remarques sur les outils de mesure.

La plupart des indicateurs de puissance présentés ci-dessus sont obtenus en exécutant des programmes de référence dont on mesure le temps d'exécution, temps donné par le système d'exploitation. Une méthode offrant plus de rigueur que le simple relevé des temps donnés par l'exécution de la charge mesurée est celle de la **double pesée**; dans un premier temps on exécute un premier programme A, saturant le processeur; on note son temps d'exécution Ea; A est ensuite relancé une seconde fois (seconde pesée) avec la charge B dont on veut mesurer le temps processeur qu'elle consomme; l'ensemble de A et B conduit à un temps d'exécution Eb, et l'allongement du temps d'exécution de A, égal à Eb-Ea, correspond au temps pendant lequel le processeur central a travaillé pour B; on parlera du "temps processeur", "temps CPU" à propos de Eb-Ea.

Malgré ses qualités, cette méthode introduit quelques biais. Entre les deux exécutions de A, le temps que le processeur consacre au système d'exploitation risque d'augmenter, en particulier dans les systèmes à temps partagé (Tome II) où le distributeur intervient plus souvent puisque l'exécution globale conduit à un plus grand nombre de tranches de temps. Lors des mesures les constructeurs peuvent aussi "optimiser" en allongeant par exemple la longueur de la tranche de temps.

Des outils **externes** au processeur permettent de réaliser des mesures **absolues**, non entachées d'erreur dans la mesure où l'outil de mesure ne perturbe pas le système mesuré. Ce sont les **logimètres matériels** (hardware monitor) qui sont capables de donner une cartographie très détaillée des adresses et des codes des instructions exécutées. Ils donnent aussi le temps pendant lequel le processeur est dans le mode maître, dans le mode esclave, etc. Ils permettent encore de préciser si le processeur, tout en paraissant actif de l'extérieur, ne perd pas en fait du temps à attendre que la mémoire centrale soit disponible. Certains ordinateurs (cas du DPS8) disposent d'instructions particulières pour synchroniser le travail du logimètre, en lui indiquant par exemple à quel instant il convient de démarrer les mesures.

Les constructeurs fournissent généralement aux utilisateurs qui le désirent, la liste des points de sonde, c'est-à-dire des signaux que le logimètre peut observer (monitor). Les grands constructeurs actuels de logimètres matériels sont TITN, TESDATA. Notons qu'en dehors des dispositifs de "capture" des signaux de l'ordinateur testé, ces moniteurs sont des ordinateurs.

chapitre 14

les principaux constructeurs

Les noms, les parts de marché des différents constructeurs n'ont pas a priori leur place dans un ouvrage sur l'architecture des ordinateurs. Il nous a cependant paru utile de leur consacrer un bref chapitre afin que le lecteur ait un premier aperçu de la réalité commerciale et industrielle du monde des ordinateurs.

14.1 – Origine des constructeurs.

Les constructeurs dominants, aussi bien du point de vue des parts de marché que de celui de l'innovation sont américains. Ils subissent actuellement au niveau technologique une forte concurrence des japonais qui fabriquent des composants estimés plus fiables et plus intégrés. Au niveau des grands et moyens ordinateurs, les européens sont à la remorque des américains et des japonais:

- Siemens en RFA dépend de Fujitsu qui lui livre des compatibles IBM, le système d'exploitation BS2000 étant propre à Siemens;
- ICL en Angleterre a du faire appel à la technologie de Fujitsu, tout en produisant une gamme d'ordinateurs avec une architecture qui lui est propre;
- Bull en France dépend d'Honeywell pour son haut de gamme DPS8, DPS88, et de plus en plus du japonais NEC, aussi bien pour le très haut de gamme DPS90 que pour sa gamme moyenne DPS7.

Le cas de Bull est intéressant pour illustrer les deux prépondérances américaine et japonaise, la seconde étant plus récente. Lorsque fut créée la CII (Compagnie Internationale pour l'Informatique) en 1964, Bull fut racheté par General Electric (GE). Quand Honeywell racheta l'activité informatique de GE vers 1970, Bull changea à nouveau de nom et d'actionnaire majoritaire, et de BGE devint CHB (Compagnie Honeywell Bull). Pour unifier les gammes issues de ces fusions et rachats, une nouvelle gamme d'ordinateurs fut conçue au début des années 1970 au sein d'Honeywell et de ses différentes filiales. Seuls les modèles français (le 64) et italien (le 62) virent réellement le jour, les ordinateurs de GE (série 6000) s'imposant aux Etats-Unis en haut de gamme. A l'époque NEC avait la licence des 6000 et des 64 qu'il fabriquait et commercialisait au Japon. NEC eut encore

la licence du successeur des 6000 (les 66). La nouvelle architecture interne des successeurs des 66 fut annoncée avec le modèle 66/85 par Honeywell mais ne put être réellement livrée en clientèle car Honeywell ne fut pas en mesure de surmonter à temps les difficultés technologiques et le manque de performance rencontrés sur les prototypes. NEC par contre réussit son 66/85 appelé S800, avec ses successeurs S900 et S1000. Ce S1000 est commercialisé sous le nom de DPS90 par Bull et Honeywell, il est plus puissant que les DPS88, successeurs réellement livrés du 66/85, mais d'architecture interne différente. Tout comme Honeywell, Bull éprouve des difficultés pour faire évoluer ses DPS7 (successeurs des 64) et doit se tourner vers NEC pour ses nouveaux ordinateurs moyens.

En haut de gamme des ordinateurs de gestion deux technologies dominent à l'heure actuelle: IBM d'une part, les japonais de l'autre. La pression japonaise devient très forte aussi dans le domaine des superordinateurs (supercomputers), c'est-à-dire des ordinateurs scientifiques dont la puissance se mesure en centaines ou milliers de MFLOPS.

Dans le domaine des **minis**, la domination américaine est encore plus importante. Ainsi Bull commercialise des MINI6 (devenus des DPS6) de conception Honeywell, des SPS9 de conception RIDGE, etc. NORSK DATA en Norvège a pu faire cavalier seul jusqu'ici et vient de s'associer avec MATRA en France.

Si l'on aborde les **micros**, IBM tient en 1984 30% du marché, Apple 10% et Commodore 6% selon IDC. Là encore nette domination des firmes américaines. En Europe Olivetti a 5% du marché européen seulement et 8% du marché italien. En Angleterre 23% du marché national sont détenus par les constructeurs nationaux ACT (les Apricot) et ACORN. Ce dernier a été repris par Olivetti en 1985.

Les tableaux 14f et 14g montrent l'importance des américains et japonais au niveau **technologique**. En 1983 70% des microprocesseurs MPU (puce contenant un processeur) vendus provenaient de fabricants américains. Quant aux MCU (puce avec processeur et avec de la mémoire) du tableau 14h, 65% viennent des japonais.

14.2 – Les principaux constructeurs.

Nous nous appuierons sur une étude publiée en juin 1985 dans la revue Datamation. On y trouve les 100 premières sociétés mondiales du secteur informatique. La première remarque qui s'impose est la **domination d'IBM**, le numéro un mondial avec un chiffre d'affaires de 46 milliards de dollars en 1985, dont 96,4% réalisés dans l'informatique, pour presque 400 000 employés. Le numéro deux, DEC, est plus de sept fois plus petit en chiffre d'affaires avec un peu plus de six milliards de dollars.

Les cinq premières sociétés sont américaines (IBM, DEC, Burroughs, CDC et NCR); elles sont au nombre de sept dans les dix premières. La première société française, Bull, est classée seizième. Outre Bull, deux firmes françaises figurent dans les cent premières sociétés: Cap Gemini Sogeti et CISI, deux sociétés de service (SSII, autrefois SSCI). La RFA est citée cinq fois (Siemens, BASF, Mannesman, Nixdorf, Triumph-Adler), l'Angleterre six fois (Scicon, Ferranti, Thorn-Emi,

Plessey, ICL, Racal), la Norvège une fois (NORSK DATA) ainsi que la Finlande (Nokia), l'Italie (Olivetti), la Suède (Ericsson), la Hollande (Philips).

Certaines sociétés dominent tous les grands secteurs de l'informatique (cas d'IBM), d'autres n'agissent que dans un segment particulier, comme Apple dans les micro-ordinateurs, STC dans les sous-systèmes périphériques.

RANG	SOCIETE	CA en 1984 M$	%/83
1	IBM	13 131	14,7
7	Sperry	1 451	11,5
3	Burroug.	1 450	11,5
6	Fujitsu	1 400	33,3
5	NCR	1 345	34,5
9	NEC	914	17,7
4	CDC	813	4,9
10	Siemens	807	17,6
12	Hitachi	772	13,6
15	Honeyw.	665	5,5

Tableau 14a: les 10 premières sociétés dans les gros ordinateurs (mainframes) selon Datamation en 1985.

RANG	SOCIETE	CA en 1984 M$	%/83
1	IBM	3 000	14,1
2	DEC	1 527	52,7
11	WANG	970	8,6
8	HP	950	29,1
19	DG	840	18,9
3	Burroug.	700	7,6
13	Olivetti	540	10,1
40	PRIME	479	15,0
45	TANDEM	477	23,1
22	Toshiba	421	11,1

Tableau 14b: les 10 premières sociétés dans les minis selon Datamation en 1985.

14.3 – Structure du marché.

Selon une étude d'IDC, le marché mondial des matériels informatiques se décompose en 1984 en quatre segments:

- 33% pour les grands systèmes. IBM tient environ 70% du marché des gros et moyens ordinateurs de gestion (45 % en 1981);
- 21,6% pour les moyens systèmes. Les superminis (350 à 700 K$ par système en 1985) représenteraient d'après INFOCORP un CA mondial de cinq milliards environ en 1984, pour un peu plus de 6000 systèmes livrés;
- 19,1% pour les petits systèmes. Selon INFOCORP le marché des petits systèmes (de 25 à 350 K$) représente en 1984 un CA mondial de 32 milliards de dollars pour un peu moins de 300 000 systèmes livrés;
- 26,3% pour les ordinateurs personnels (PC). D'après le Yankee Group, en 1990 5,2 millions de micros seront vendus (950 000 en 1984) pour 13,4 milliards de dollars (11,5 selon IDC). Ce marché dépasserait en valeur les autres marchés cités ci-dessus dès 1984: une comparaison d'IDC en MIPS installés aux Etats-Unis en 1984 donne environ 50 000 MIPS pour les moyens et gros systèmes du type 3030, 3080, 3090, 43XX et plus de 500 000 MIPS pour les PC IBM. Si les micros 8 bits (type Apple II) représentent encore 50% des ventes en 1984, ils auront disparu en 1990 selon IDC.

RANG	SOCIETE	CA en 1984 M$	%/83
1	IBM	4 000	53,8
14	Apple	1 898	74,9
23	Commod.	1 130	21,8
8	HP	510	27,7
7	Sperry	503	30,2
38	TANDY	403	-29,1
57	Converg.	362	121,6
61	Compaq	329	195,8
13	Olivetti	290	14,8
9	NEC	259	30,0

Tableau 14c: les 10 premières sociétés dans les micros selon Datamation 1985.

RANG 81	RANG 82	SOCIETE	CA en 1981 EUROPE	MONDE
1	1	IBM	8 846	29 070
2	3	Bull	1 311	1 352
3	2	Siemens	1 296	1 330
4	8	DEC	1 162	3 198
5	4	ICL	1 067	1 513
6	5	Olivetti	1 006	1 436
7	6	Sperry	850	3 039
8	9	CDC	765	3 101
9		Philips	750	
10	10	Burroug.	742	3 045
11	7	NCR	728	2 838
12	11	Nixdorf	678	2 856
13	12	HP	604	1 771

Tableau 14d: les constructeurs en Europe selon Datamation 1982.

D'après une enquête récente de Datapro (cf. Computerworld du 15 juillet 1985 page 1) portant sur 937 utilisateurs, l'âge moyen des ordinateurs installés était de dix mois.

ANNEE	IBM	APPLE	TANDY	HP	DEC	TOTAL
1981	20 6,7%	160 54,2%	90 30,5%	25 8,5%		295
1982	180 33	230 42	85 16	40 7	10 2%	545
1983	420 37	430 37,8	140 12,3	60 5,3	85 7,5%	1 135

Tableau 14e: production d'ordinateurs personnels aux USA en milliers d'unités et en pourcentage de la production totale.

Une autre segmentation du marché, non pas basée sur la taille des systèmes, mais sur leurs applications, conduit à évaluer:

- le marché des machines à **tolérance de panne** (cf. chapitre 19) à 16 milliards de dollars en 1985 selon R. Morris (ex patron de CDC). Soulignons que ce marché, comme d'autres, a été surestimé plusieurs fois. Une étude IRD de 1984 l'évaluait à moins de un milliard de dollars en 83 et l'estimait à 4,2 milliards en 1987; Frost et Sullivan l'estiment à 162 millions de dollars en 1984 en Europe, soit 1 % du marché, et à 1900 millions en 1990 (3 % du marché).
- le marché des **superordinateurs** croît selon Amdahl de 40 % par an en moyenne pour atteindre 1 000 à 2 000 installations en 1990.

Les sociétés qui jouent un rôle dans le domaine des ordinateurs sont bien sûr les fabricants, les revendeurs, les distributeurs, mais

aussi les sociétés de service, les sociétés de conseil, les groupes d'utilisateurs, les pouvoirs publics, etc.

Les constructeurs classiques conçoivent généralement les architectures externe et interne, ils en réalisent le système d'exploitation (Tome II) et commercialisent eux-mêmes leurs produits. Ils ne conçoivent habituellement pas les circuits intégrés qui forment leurs briques de base et qu'ils achètent auprès des "fondeurs" de silicium comme Texas Instruments, Intel, NEC, etc.

Certains de ces constructeurs conçoivent et fabriquent tout ou partie des circuits intégrés qu'ils utilisent. IBM en est un exemple et de plus a récemment pris une participation au capital d'Intel (cf. tableau 14f). NEC quant à lui fabrique tout. De même des fabricants de circuits intégrés vendent leurs propres micro ou mini-ordinateurs: Intel, Texas Instruments, etc.

RANG 79	RANG 83	RANG 84	RANG 85	SOCIETE	CA en M$
1	1			Texas I.	1 276
2	2			IBM	1 262
3	3			Hitachi	958
7	4	3	1	NEC	942
5	5			Motorola	842
4	6		6	Philips	805
6	7			NS	783
10	8			Fujitsu	692
8	9			INTEL	655
11	10			Toshiba	597

Tableau 14f: les dix premiers producteurs de VLSI en 1983.

RANG	SOCIETE	PART MARCHE 83	PART MARCHE 80
1	NEC	12,5	8,0
2	INTEL	11,8	12,6
3	ZILOG	10,1	11,5
4	Mostek	9,3	5,0
7	Motorola	5,5	12,7

Tableau 14g: les fabricants de microprocesseurs MPU selon Dataquest (CA de 320 millions de $ en 1983 pour 75 millions de pièces vendues, dont 90% de microprocesseurs 8 bits).

14.4 – Ventes indirectes.

Les constructeurs ne vendent pas toujours directement les produits qu'ils fabriquent. D'autres types de sociétés, après un éventuel ajout de fonctionnalités ("valeur ajoutée"), revendent ces produits. On trouvera:

- des sociétés de service, créées depuis les années 50,
- des OEM (original equipment manufacturer). Le terme provient de l'automobile et est apparu dans l'informatique dans les années 60 avec les PDP-8 et PDP-11. Un OEM est une société qui achète un grand nombre d'éléments, les "intègre", ajoute de la valeur et revend le produit à sa clientèle,
- les fournisseurs de "systèmes clé en main" configurent des systèmes dont le matériel et le logiciel sont déjà installés,
- les intégrateurs de systèmes,
- de nouveaux agents au nom basé sur "valeur ajoutée" sont apparus plus récemment: VAR (value added remaker), VAD (value added dealer),
- etc., [FRE85] note que la définition de ces termes varie d'un fournisseur à l'autre.

Selon IDC le pourcentage des livraisons de matériel informatique effectuées par les canaux indirects va de 19 à 33 % selon le type de matériel en 1983.

RANG	SOCIETE	PART MARCHE 83	80
1	NEC	20,4	15,0
2	TOSHIBA	11,1	
3	NSC	9,6	
4	MATSUSH.	9,3	
5	HITACHI	9,3	
6	MOTOROLA	6,5	
7	INTEL	6,2	
	TEXAS	5	37,4

Tableau 14h: les fabricants de microprocesseurs MCU (PC, MEM, MEV sur une même puce) selon Dataquest (CA de 530 000 000$ en 1983 pour 260 000 000 pièces vendues).

RANG VAL.	SOCIETE	SYSTEME	NOMBRE
1	IBM	3081	1 528
2	DEC	VAX11/780	12 620
5	IBM	4341	6 373
7	IBM	S/34	54 211
8	IBM	S/38	7 974
10	DEC	VAX11/750	10 205
11	HP	3000	13 508
12	IBM	S/36	15 100

Tableau 14i: nombre de systèmes installés aux Etats-Unis au 1-1-85 selon Computer Intelligence Corp. Classement en valeur.

chapitre 15

historique et évolution

Il serait incomplet de parler d'architecture des ordinateurs sans donner une idée des "générations" d'ordinateurs et sans aborder les grandes lignes de leur évolution.

15.1 – Les générations.

Les ordinateurs existent depuis une quarantaine d'années et on leur attribue communément quatre générations successives, sans compter une préhistoire et des discussions sur la future cinquième génération.

Ce qui sépare le plus une génération de sa suivante est l'évolution de la technologie. C'est d'ailleurs ce qui est le plus visible pour le plus grand nombre puisque le volume apparent de la machine s'en ressent.

- la "préhistoire" est caractérisée par les relais électromécaniques et correspond à la période allant jusqu'en 1950;
- les relais, lampes et tubes sont utilisés par la première génération qui mène à l'année 1958 environ;
- avec le transistor apparaît la deuxième génération. Il avait été découvert en 1948 par Bardeen, Brattain et Schockley de la Bell Telephone;
- les années 1964 à 1970 correspondent à peu près à la troisième génération qui voit l'explosion des circuits intégrés inventés par Jack Tilby chez Texas Instruments en 1958;
- la quatrième génération est celle de l'intégration à grande échelle (LSI) et des microprocesseurs;
- la cinquième génération devrait concrétiser la synthèse des domaines suivants:
  * l'intelligence artificielle (IA, AI en américain) dont l'un des éléments les plus étudiés actuellement est celui des **systèmes experts**, basés sur la connaissance (knowledged based expert system). On espère ainsi passer du traitement actuel de l'information au traitement de la "connaissance" (passage de l'informatique à la connaissique?);
  * les langages de programmation de très haut niveau permettant à l'utilisateur d'exprimer ce qu'il veut (le **"quoi"**) sans l'obliger à indiquer à l'ordinateur **comment** il doit procéder pour lui four-

nir les résultats attendus. Ce sera le passage des langages procéduraux aux langages naturels;
* l'informatique décentralisée;
* l'intégration à ultra grande échelle sur le plan technologique (ULSI).

Ils doivent de plus offrir des entrées/sorties basées sur la voix, l'image, le graphique, démarche cohérente avec les langages naturels.

Les langages, les autocommutateurs et nombre d'autres produits, font aussi l'objet d'un découpage en générations que nous verrons dans le Tome V.

15.2 – La préhistoire (avant 1950).

Les ordinateurs marquant cette période sont l'ENIAC de Mauchly (1946), les MARK I et II de Aiken, construits en 1944, le WWI (Whirlwind) de Von Neumann au MIT (1947) à mots de 16 bits avec un langage de programmation symbolique alors que la caractéristique des machines de cette période est la programmation en langage machine; l'EDVAC date de 1947, l'IBM 704 dès 1948, l'EDSAC en 1949 utilisent des lignes à retard et disposent du "programme enregistré", notion datant de 1945.

Ces ordinateurs ne font pas encore l'objet d'une commercialisation. Ce sont des prototypes, des projets de recherche, militaires ou universitaires. La vitesse de traitement atteint 300 multiplications par seconde, 5 000 additions par seconde (avec des nombres de 12 chiffres sur l'ENIAC), 20 000 opérations par seconde sur la WWI.

15.3 – Première génération (1950 à 1958).

En dehors de l'évolution technologique déjà citée, la première génération correspond à la généralisation des langages d'assemblage, l'apparition des précurseurs des langages évolués: Mathmatic préfigure l'Algol, Flowmatic le Cobol. L'IBM 709 est doté d'un compilateur, les sous-programmes se généralisent, l'IBM 702 dispose d'un macro-assembleur, l'IBM 701 a un chargeur absolu.

Remington Rand Corp. (devenu Sperry depuis), avec Mauchly et Eckert, commercialise en 1951 le premier ordinateur commercial, l'UNIVAC 1 (UNIVersal Automatic Computer) qui est une machine décimale à mots de 12 chiffres, avec dérouleur de bande magnétique et qui atteint 1000 opérations par seconde. Il fut construit pour le bureau du recensement. Les machines de cette période sont spécialisées:
- machines de **gestion**: IBM 702 (1952) concurrent de l'UNIVAC 1, IBM 705 (1954), l'UNIVAC II en 1956,
- machines **scientifiques**: IBM 701 en 1952 avec mots de 36 bits et mémoire de 2 Kmots, UNIVAC 1103A à mots de 36 bits aussi en 1955, avec instructions à deux adresses et avec interruption, IBM 704 en 1956 avec 4 Kmots de mémoire, 3 registres d'index, flottant câblé et adressage indirect.

En dehors des bandes (UNIVAC 1), les tambours font leur apparition sur l'IBM 604 en 1948. Les mémoires commencent à être constituées par des tores (IBM 704 et 705 en 1954). Après avoir fait passer les

entrées/sorties par l'accumulateur, leur traitement commence à être distribué en dehors du processeur avec des accès à la mémoire partagés entre le processeur et le canal sur l'IBM 709 en 1958.

La deuxième génération correspond encore à l'apparition des grands fabricants d'ordinateurs: UNIVAC, Bull (Gamma 3 en 1952), Burroughs (UDEC en 1953), CDC fondé en 1957 par des transfuges d'UNIVAC, Raytheon, Honeywell, RCA, General Electric, NCR, etc.

15.4 – La seconde génération (1958–1964).

Caractérisée par le transistor, cette génération correspond aussi aux premiers langages évolués, aux simultanéités entre l'activité du processeur central et les entrées/sorties, aux utilitaires.

L'un des premiers ordinateurs transistorisés fut le 501 de RCA en 1959; il disposait d'un compilateur Cobol. Le NCR-GE304 est en 1959 le premier ordinateur entièrement transistorisé avec 800 diodes et 4000 transistors dans le processeur; l'addition nécessite 60 microsecondes.

En 1960 IBM sort sa première machine transistorisée, le 7070. La même année DEC commercialise le PDP-1 et UNIVAC produit deux exemplaires d'un superordinateur, le LARC, avec un processeur central opérant sur du flottant en BCD, avec pipeline et mémoire centrale entrelacée (chapitres 17 et 18), ainsi qu'un processeur d'entrées/sorties disposant de sa propre mémoire.

1959 voit la 1620 d'IBM avec des caractères 6 bits (sextets); 2000 furent vendues. L'IBM 1401 (1959) fut vendue à 20 000 exemplaires; elle avait des caractères de 7 bits occupant en fait un octet, le huitième bit servant à délimiter les chaînes de caractères; la longueur des chaînes n'avait donc pas à être codée dans l'instruction. La H200 d'Honeywell offrait une compatibilité ascendante avec la 1401. 1961 est l'année du STRETCH (ou 7030), projet ambitieux d'IBM qui n'offrit pas les performances attendues, mais permit d'expérimenter plusieurs dispositifs architecturaux: exécution simultanée de plusieurs instructions, deux registres limite protégeant les programmes en multiprogrammmation.

L'ATLAS dispose en 1962 de la pagination (Tome II). Le 5000 de Burroughs orienté pile, avec adressage à base de descripteur et segmentation est conçu pour traiter efficacement l'Algol. Plutôt lent il est remplacé en 1964 par le B5500.

15.5 – La troisième génération (1964–1970).

Avec l'apparition des circuits intégrés, cette génération voit l'affirmation des machines universelles (à vocation gestion et scientifique) et des gammes d'ordinateur. La microprogrammation est de plus en plus utilisée au cours de cette période. La périphérie voit apparaître les piles de disque (diskpack, cf. Tome IV), les systèmes d'exploitation se stabilisent et offrent multiprogrammation et temps partagé (Tome II).

La gamme 360 en 1964 concrétise la plupart de ces caractéristiques, relayée par la série 370 qui popularise la pagination. Quelques constructeurs commencent à éprouver des difficultés. Ainsi Bull devient BGE en 1964 et doit abandonner ses propres gammes pour commer-

cialiser celles de GE. En 1969 Xerox rachète SDS, le concepteur des SIGMA vendues sous le nom de 10070 par CII; Honeywell rachétera SDS, appelé alors XDS, au début des années 80.

Les minis voient le jour dès le début de cette période. DEC annonce les PDP-8 en 1965, le PDP-11/20 en 1970.

15.6 – La quatrième génération (1970 et au-delà).

Certains la font débuter en novembre 1975 avec l'annonce du 470/V6. C'est la génération de l'évolution très rapide de la micro-électronique avec l'intégration croissante, le phénomène microprocesseur. C'est aussi un énorme effort dans les systèmes d'exploitation qui deviennent de véritables monstres de plusieurs millions de lignes source et consacrent la tendance du mode dialogué (l'interactif) amorcée à la fin de la troisième génération, rodé sur les minis dotés du Basic, expérimenté sur de gros systèmes comme MULTICS ou CP/CMS.

La quatrième génération voit apparaître de plus en plus de multiprocesseurs, le D-825 de Burroughs de 1962 étant considéré comme le premier vrai quadriprocesseur avec seize modules mémoire et une matrice de connexion (crossbar, cf. chapitre 17).

15.7 – L'évolution.

L'évolution des sous-systèmes centraux se traduit par:

- une réduction de la **taille** provoquée par la formidable progression de la technologie des semi-conducteurs. La surface d'un transistor a ainsi été réduite par un facteur supérieur à 10 000 depuis 1950 selon [LAZ85]. Entre l'ENIAC de 1946 et le F8 de Fairchild en 1974, le volume a été divisé par 300 000 pour une puissance à peu près équivalente (200 microsecondes pour une addition de 12 chiffres sur l'ENIAC, 150 pour une addition de 8 chiffres sur le F8 selon [HOG77]);
- une diminution importante de la **puissance consommée**: 56 000 fois moins pour le F8 sur environ trente ans;
- une augmentation continue de la **puissance de calcul** grâce à la vitesse accrue des semi-conducteurs ainsi qu'aux améliorations architecturales. Cette puissance est de moins en moins chère (cf. tableau 13b);
- une croissance continue des **tailles des mémoires** centrales qui ont plusieurs centaines de milliers d'octets sur les micros, plusieurs millions sur les minis, plusieurs dizaines de millions sur les gros ordinateurs de gestion. Cette augmentation de la taille des mémoires configurées va de pair avec celle de l'espace adressable: la série 370 d'IBM est ainsi passée à un "adressage étendu" (XA comme eXtended Architecture) afin que les programmes passent des 16 Mo devenus trop petits (adressage par 24 bits d'adresse) à 2 Go (adressage par 31 bits);
- la logique devenant moins chère et plus facilement manipulable à un haut niveau de fonctionnalité, la **décentralisation** s'accroît sans cesse, la fonction d'unité de commande se retrouve à n exemplaires dans l'ordinateur: dans le (ou les) processeur, dans la mémoire qui est dotée d'un ou plusieurs contrôleurs (mot utilisé pour unité de

commande dans la pratique dès que l'on sort du sous-système central), dans les processeurs d'entrée/sortie qui déchargent le processeur central des tâches de gestion physique de ces entrées-sorties comme nous le verrons dans le Tome III. Les sous-systèmes périphériques deviennent eux-mêmes de plus en plus "intelligents", c'est-à-dire qu'ils savent faire de plus en plus d'actions que le sous-système central n'a plus à réaliser;
- le nombre de dispositifs utilisant les techniques de **parallélisme** et de **pipelining** afin d'accroître les performances;
- une **standardisation** croissante vers les machines de type 370 dans les moyens et gros ordinateurs. Le pourcentage de ces machines déroulant des instructions 370 croît sans cesse. Le phénomène microprocesseur crée aussi une standardisation dans les plus basses puissances, et en dehors des architectures RISC, on ne voit plus apparaître de nouveaux jeux d'instructions;
- l'**universalité** des domaines d'application: les ordinateurs pénétrent une grande quantité de secteurs d'activité, le bureau avec la bureautique (office automation), le commerce de détail, les professions libérales, les ateliers industriels, etc.;
- la fiabilité et la **disponibilité** ont beaucoup progressé.

15.8 – Le phénomène microprocesseur.

Le microprocesseur n'est pas né volontairement et le premier commercialisé (le 4004 d'INTEL, en 1971) n'a reçu ce nom qu'un an après sa naissance, en 1972.

En 1968, Viatron annonça un système de gestion loué à 40$ par mois (Electronics, 14 octobre 1968, p. 193) qui contenait un microprocesseur 8 bits; mais en 1970, Viatron disparaissait.

A la même époque, General Electric cherchait à concevoir un circuit intégré pouvant être reconduit (re-programmé) sur ses différents terminaux. C'est ainsi que fut créée une unité logique de 8 bits appelée BLU (Basic Logic Unit), utilisée avec différents types de terminaux. C'est essentiellement ce qui fut fait avec les premiers microprocesseurs. L'idée était dans l'air.

A l'origine du 4004 on trouve un fabricant de calculettes japonais, Busicom, qui demande en 1969 à INTEL de lui fournir des circuits intégrés pour une nouvelle calculette. Afin d'obtenir une solution optimale, au lieu de réaliser des circuits classiques (du "câblé", random logic en américain), INTEL produit le MCS-4, une puce de 2.300 transistors et 16 broches qui deviendra le 4004 (cf. [GAR85]).

CTC, un fabricant de caisses enregistreuses, sera à l'origine du 8008, en avril 1972, avec 18 broches.

Le phénomène microprocesseur démarre alors : Rockwell sort le PPS-4 en 1972, suivi par d'autres. En avril 1974 INTEL annonce le 8080 en NMOS, alors que les précédents étaient en PMOS; le 8080 a 5.000 transistors et 40 broches.

Les microprocesseurs 16 bits annoncés en 1974 se répandront réellement en 1978 avec le 8086.

Sur le plan technologique, la tendance actuelle est au CMOS et ses dérivés.

chapitre 16

parallélisme et anticipation

La vitesse de propagation des signaux est de 300 000 km/s (vitesse de la lumière) ou 30cm/ns et quelle que soit la technologie, il y a donc toujours un seuil en dessous duquel on ne peut pas descendre.

Les performances d'un ordinateur sont conditionnées par deux grands facteurs:
- la vitesse de la technologie utilisée,
- l'architecture.

Nous avons vu jusqu'ici les architectures simples, séquentielles, dans lesquelles il n'y a pratiquement pas de parallélisme de traitement au-delà d'un niveau assez bas qui est celui des microtransferts.

Pour accroître les performances, les architectes d'ordinateur ont imaginé des structures faisant appel:
- au **parallélisme**, c'est-à-dire à l'exécution simultanée d'actions **identiques**,
- à **l'anticipation** ou pipelining qui conduit à dérouler simultanément des actions **différentes**.

16.1 – Le parallélisme.

Considérons un circuit réalisant une fonction F (figure 16a) avec une entrée E et une sortie S. Une structure parallèle **dupliquera** ce circuit en n copies identiques, disposant chacune d'entrées Ei et de sorties Si (figure 16b).

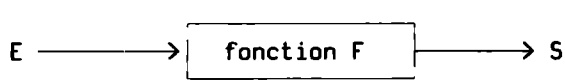


Figure 16a: circuit réalisant la fonction F.

Une autoroute offre une analogie intéressante pour illustrer le parallélisme: la performance y est mesurée par son débit, le nombre de voitures y passant par unité de temps. Le débit sera plus important si au lieu d'une voie on en met deux ou même trois. Deux voies doubleront le trafic unitaire, trois voies le tripleront .... ou presque:
- il faut qu'il y ait suffisamment de trafic pour utiliser les ressources disponibles et ainsi augmenter le débit,

- les trafics des différentes voies ne sont pas totalement indépendants, il y a des interactions qui diminuent la performance globale,
- les dispositifs d'accès peuvent se révéler insuffisants pour charger pleinement l'autoroute, les sorties sont susceptibles de provoquer des ralentissements.

Nous verrons qu'il en est de même dans les ordinateurs. [KLE85] présente un développement sur ce problème de "facteur d'accélération" dans les multiprocesseurs.

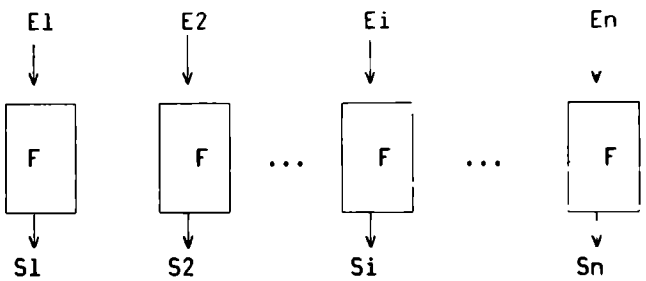


Figure 16b: duplication d'un même circuit dans les solutions parallèles.

16.2 – L'anticipation.

Reprenons le circuit réalisant la fonction F introduite ci-dessus. L'anticipation ou **recouvrement** conduit à **décomposer** la fonction en sous-fonctions successives. Tout comme les n circuits dupliqués dans le cas du parallélisme, les n circuits obtenus pour réaliser les n sous-fonctions travailleront simultanément sur n informations différentes. La différence réside dans le fait que dans le parallélisme les n informations en sont au même stade d'avancement dans chacun des n circuits, alors qu'avec l'anticipation les n informations en sont toutes à un état d'avancement différent.

La figure 16c montre un pipeline dans lequel la fonction F a été décomposée en quatre sous-fonctions SFi. Le circuit global présente donc quatre circuits que l'on appellera des **étages**. Chaque étage attend que le précédent ait élaboré son résultat pour le prendre comme entrée. La figure 16c illustre deux des trois états d'un pipeline:

- le **remplissage**: la première information entre dans le pipeline à l'instant t1, mais le premier résultat ne sort du pipeline qu'à l'instant t5. De t1 à t4 seule une partie du pipeline est active puisque tous les étages ne sont pas sollicités. On dira que le pipeline est en train de se remplir; plus il y a d'étages et plus le pipeline est long à se remplir;
- le régime de **croisière** lorsque le pipeline est plein et travaille à plein régime;
- le **vidage**, non représenté sur cette figure (voir la figure 16d), est l'inverse du remplissage. Quand il n'y a plus d'informations à l'entrée du pipeline, les premiers étages n'ont plus rien à faire, le pipeline se vide peu à peu.

L'étage le plus lent fixe le débit du pipeline en régime de croisière.

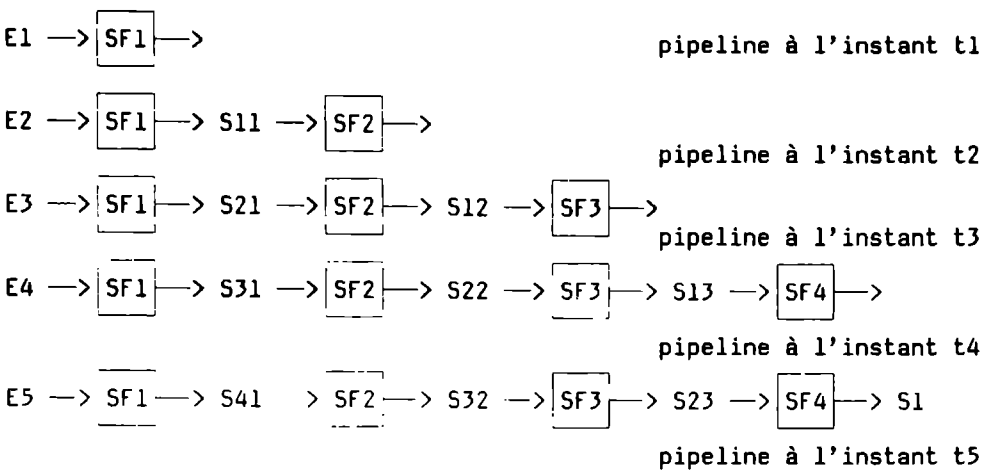


Figure 16c: pipeline à 4 étages. Sij est le résultat intermédiaire élaboré par le circuit SFj à partir de l'entrée Ei.

On estime que sur le CYBER 205 de CDC il faut des vecteurs d'au moins 200 éléments pour atteindre la moitié de la vitesse de croisière. D'où les plages de chiffres concernant les MFLOPS dans le tableau 13f. L'image du pipeline est la chaîne de montage automobile.

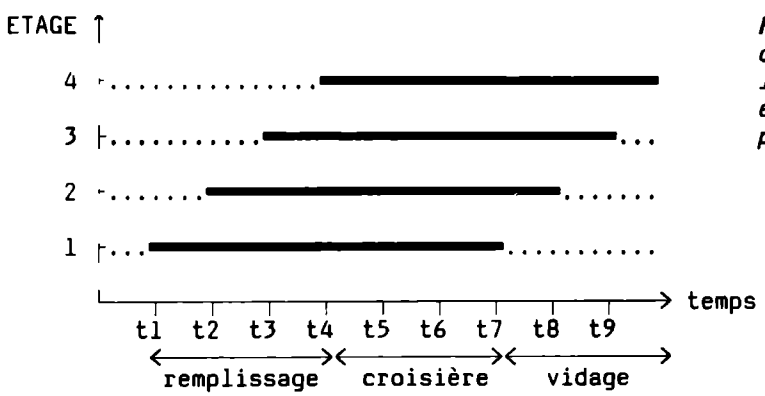


***Figure 16d:** diagramme de l'occupation des étages d'un pipeline.*

Pour comparer les deux mécanismes de parallélisme et d'anticipation supposons que les circuits F et SFi réalisent tous leur fonction en un temps T. Dans la solution basée sur le parallélisme, les n circuits F fourniront n résultats à chaque période T alors que dans la solution pipeline on aura un résultat à chaque période T durant le régime de croisière. Pour que les deux solutions soient identiques sur le plan des performances pendant le régime de croisière, il faudrait que chacun des p circuits SFi soit n fois plus rapide que le circuit F.

Sur le plan des coûts la solution pipeline est plus avantageuse puisqu'il n'y a pas duplication totale des ressources.

Les pipelines font l'objet de nombreuses études et réalisations. On distingue généralement les pipelines mono et multifonctions selon qu'ils réalisent toujours la même fonction ou peuvent être "paramétrés" pour modifier la fonction. Parmi les multifonctions on distingue encore les pipelines statiques et dynamiques, ces derniers se prêtant mieux à des changements fréquents.

En ce qui concerne le vocabulaire, on parlera aussi d'unités **segmentées** (segmented) pour dire "pipelinées".

Différentes unités pipelinées peuvent être **chaînées** (chained) pour former une super-unité : les résultats élaborés par une unité A forment directement les entrées d'une unité B, comme dans un pipeline, sans avoir à être stockés de façon intermédiaire dans des registres ou dans la mémoire. C'est le cas des CRAY.

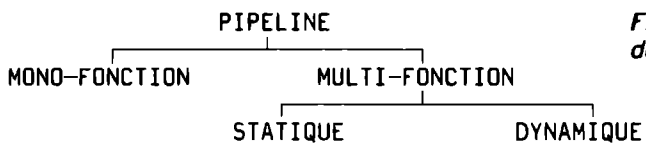


Figure 16e: types de pipelines.

16.3 – Problèmes posés.

Dans un processeur séquentiel toutes les opérations d'une instruction sont terminées avant le début de l'instruction suivante. L'ordre réel de l'exécution des instructions correspond exactement à l'ordre prévu par le programmeur.

Il n'en est plus de même lorsque les mécanismes de pipeline et de parallélisme sont mis en oeuvre. Des conflits apparaissent:

- conflits **structurels** lorsque deux instructions essaient d'utiliser simultanément un même opérateur ou le même étage d'un pipeline,
- conflits liés aux **données** lorsqu'une zone de stockage est accédée ou modifiée par deux instructions différentes. Les instructions Ii, Ij, Ik étant écrites dans cet ordre par le programmeur:
  * l'instruction Ij doit attendre la fin de l'exécution de Ii avant de lire la donnée modifiée par Ii,
  * l'instruction Ik modifie une donnée utilisée par Ij et doit donc attendre la fin de l'utilisation de cette donnée par Ij avant d'en modifier la valeur,
  * etc.

chapitre 17

parallélisme et anticipation au niveau du processeur

17.1 – Parallélisme.

Lorsque l'on aborde le parallélisme au niveau du processeur central, il faut distinguer deux types de parallélisme:

- un parallélisme **externe** qui conduit à mettre en oeuvre plusieurs processeurs centraux. Ces processeurs peuvent de plus appartenir:
  * au même ordinateur, ou
  * à différents ordinateurs.

  Dans le premier cas on parlera de **multiprocesseurs** (cas des DPS8, des Sperry 1100, etc.), dans le second on parlera de **multisystèmes** et il y aura de nombreuses variantes selon que les processeurs sont reliés par canal, sous-système périphérique disque, réseau local, réseau général, etc. La frontière entre multiprocesseurs et multisystèmes relève parfois d'un certain flou. Les deux cas font partie de ce que l'on appelle le **multitraitement**, mot dans lequel on retrouve le terme de processeur en français (process en américain équivaut à traiter).

  IBM utilise une terminologie différente: les systèmes à processeurs "fortement couplés" (tightly coupled) correspondent à nos multiprocesseurs, alors que les systèmes "faiblement couplés" (loosely coupled) correspondent à certains de nos multisystèmes.
- le parallélisme **interne** au processeur se concrétise par la présence de:
  * plusieurs **opérateurs** mis en parallèle;
  * plusieurs **unités** de traitement de l'instruction dupliquées. C'est le cas de l'AS9000, un compatible IBM de NAS, qui dispose de deux unités réalisant en parallèle la phase de chargement (fetching) des instructions;
  * de **bus** multiples. Chaque opérateur peut ainsi disposer de ses propres bus pour ses entrées et ses sorties.

17.2 – Anticipation.

Le pipelining ou anticipation dans les processeurs se retrouve dans:

- les opérateurs,
- le traitement des instructions,
- l'accès aux données,

les deux derniers cas correspondent en fait à l'accès à la mémoire centrale et ils sont étudiés totalement ou de façon complémentaire dans le chapitre suivant. C'est dans le contexte de l'anticipation que nous étudierons les antémémoires et les tampons d'instruction.

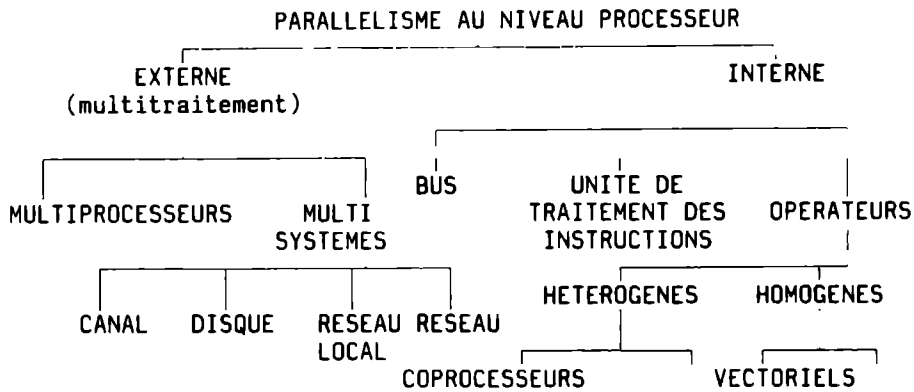


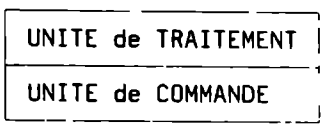
Figure 17a: classification des parallélismes au niveau processeur.

17.3 – Parallélisme au niveau des opérateurs.

Ce parallélisme se présente sous trois formes:

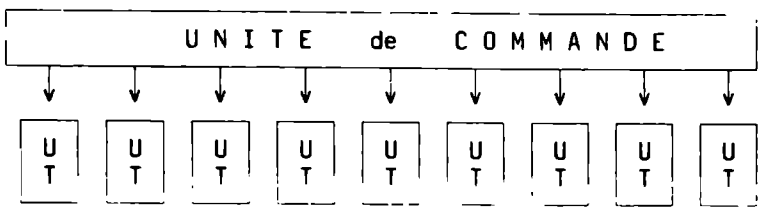
- lorsque les opérateurs en question sont **hétérogènes** (cas des CYBER et DPS88 que nous avons vus précédemment par exemple) on se trouve dans une situation où plusieurs instructions se déroulent dans le processeur, chacune sollicitant des opérateurs différents durant leur phase d'exécution. Si deux instructions ont besoin du même opérateur, leur exécution est sérialisée. Le processeur doit aussi tenir compte des cas où une instruction nécessite la fin de l'exécution d'une instruction précédente avant de pouvoir se terminer; c'est par exemple le cas d'une instruction utilisant un adressage indexé par X alors qu'une instruction précédente modifiant X n'est pas terminée;
- lorsque les opérateurs sont **homogènes**, on a généralement affaire aux instructions **vectorielles** que nous avons citées dans un chapitre précédent. On parlera aussi de machines **UIMD** (flot unique d'instructions, flot multiple de données, SIMD en américain selon la célèbre typologie de Flynn) lorsque c'est toute l'unité de traitement (opérateur et registres de données) qui est dupliquée;
- nous citerons ici les **coprocesseurs** comme l'INTEL 80287 ou le NS32081, bien qu'ils aillent au-delà de la notion d'opérateur.

Nous ne parlerons pas par contre des processeurs vectoriels (de FPS, de Warrior qui suit la norme IEEE-P754, etc.) qui se connectent sur l'ordinateur en tant que sous-système périphérique.



a) PROCESSEUR CENTRAL CLASSIQUE

Figure 17b: structur d'une machine UIMD.



UT: unité de traitement

b) MACHINE UIMD

Si l'on fait intervenir la mémoire avec une distribution sous forme de mémoire locale au niveau de chaque unité de traitement, on obtient la structure de la figure 17c. Nous n'étudierons pas plus avant les réseaux d'interconnexion dans ce tome car ils nous conduiraient trop loin. Indiquons simplement que ces réseaux d'interconnexion peuvent être matriciels (d'où le nom de machines **cellulaires**, machines **tableaux**, tableaux de processeurs, etc.), chaque UT de la matrice traitant un point dans le cas d'un traitement d'image par exemple. C'est dans cette catégorie que l'on trouvera aussi les structures **systoliques**. Ces structures à deux dimensions (2D) peuvent d'ailleurs évoluer vers des structures à trois dimensions (3D): projet SPHINX d'Orsay visant à mettre au point une machine pyramidale avec une structure UIMD à l'intérieur d'un étage, et MIMD entre étages.

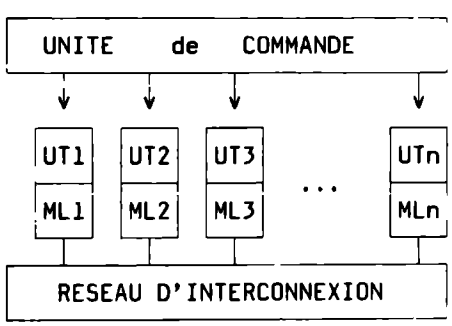


Figure 17c: structure d'une machine UIMD avec mémoires locales.

UT : unité de traitement
ML : mémoire locale

Dans la catégorie des machines UIMD on trouve les:

- ILLIAC IV, machine conçue à la fin des années 60 par l'université de l'Illinois et réalisée par Burroughs (64 UT sur des mots de 64 bits ou 128 UT sur des mots de 32 bits);
- DAP d'ICL avec 64 x 64 UT, chacune n'opérant que sur un seul bit. Dans une configuration 32 x 32 sa puissance est estimée à environ 36 MFLOPS sur des nombres de 32 bits.

Chaque UT possède un additionneur (sur un bit), 3 registres de un bit pour stocker la somme, le report, l'activité, 16 Ko de mémoire locale. Chaque UT peut de plus référencer les mémoires locales de ses quatre voisins nord, sud, est et ouest;
- STARAN et MPP de Goodyear, avec 132 x 128 UT pour le MPP, dont 128 x 128 utilisables à un moment donné. L'UT est analogue à celle du DAP, mais plus complexe du fait de la présence d'un registre à décalage programmable;
- PHOENIX avec 1024 UT;
- MASF de Burroughs avec 512 UT;
- CLIP4 de l'University Colledge de Londres avec 96 x 96 UT;
- l'AAP de NTT au Japon, etc.

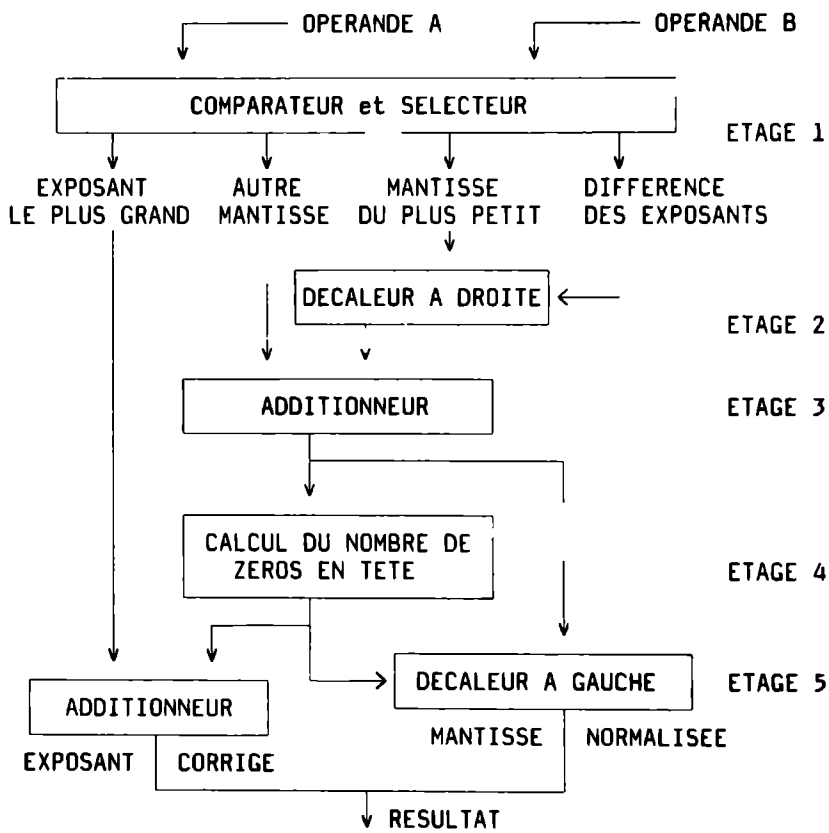


Figure 17d: additionneur à 5 étages.

17.4 – Anticipation dans les opérateurs.

Que le processeur ait un ou plusieurs opérateurs, homogènes ou hétérogènes, ces derniers peuvent être pipelinés (cas du CYBER 205) ou non. L'opérateur pipeliné correspond tout à fait à ce que nous avons présenté dans le chapitre précédent au sujet de l'anticipation. La

figure 17d schématise un opérateur d'addition en flottant avec 5 étages.

Le premier étage compare les deux opérandes et choisit (fonction "sélecteur" dans la figure 17d) le plus grand, ce qui oblige (rôle du second étage) à diminuer la mantisse du plus petit opérande, afin que toutes deux soient alignées sur le même exposant et puissent ainsi être additionnées, ce que fait le troisième étage. Afin de normaliser le résultat dans le cinquième étage, il faut voir (rôle du quatrième étage) s'il n'y a pas des zéros en tête de la mantisse somme. Le cas inverse, où la mantisse est supérieure à un, à la suite de l'addition (cas de 0,5 + 0,7 par exemple) doit aussi être envisagé, et c'est le rôle de l'additionneur du cinquième étage. Les opérateurs du CRAY 1 ont de deux (cas de l'opérateur logique) à dix-sept étages.

17.5 – Pipelining dans le traitement des instructions.

Si les opérateurs qui traitent les données et les adresses peuvent ainsi être pipelinés, les circuits qui déroulent les instructions peuvent l'être de façon analogue.

Nous avons déjà introduit deux phases dans le déroulement d'une instruction. Une solution pipeline conduit dans ce contexte à avoir à un instant donné deux instructions en cours de traitement dans le processeur:
- l'une est dans sa phase d'exécution,
- l'autre est dans sa phase de chargement.

L'unité de commande apparaît alors comme composée de deux sous-unités que nous appellerons:
- unité d'instruction (UI) pour le chargement de l'instruction suivante,
- unité d'exécution (UE) pour la phase d'exécution. On range dans l'UE l'unité de traitement UT puique c'est pendant la phase d'exécution que l'on fait appel aux opérateurs.

En première approximation, l'UI travaille sur l'instruction, alors que l'UE travaille sur les opérandes des instructions.

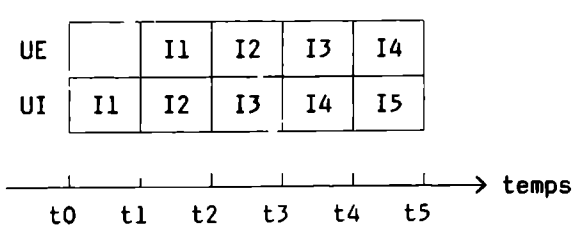


Figure 17e: contenu des unités d'instruction et d'exécution en fonction du temps.

Ce découpage en deux phases, qui existe sur le 6502, peut se généraliser à trois, quatre (cas de l'iAPX286) et plus. On trouve sur certains processeurs jusqu'à six phases avec les circuits correspondants:
- phase d'appel de l'instruction,
- phase de décodage,

- phase de calcul de l'adresse effective de l'opérande,
- phase d'appel des opérandes,
- phase d'exécution de l'opération,
- phase de calcul de l'adresse de l'instruction suivante.

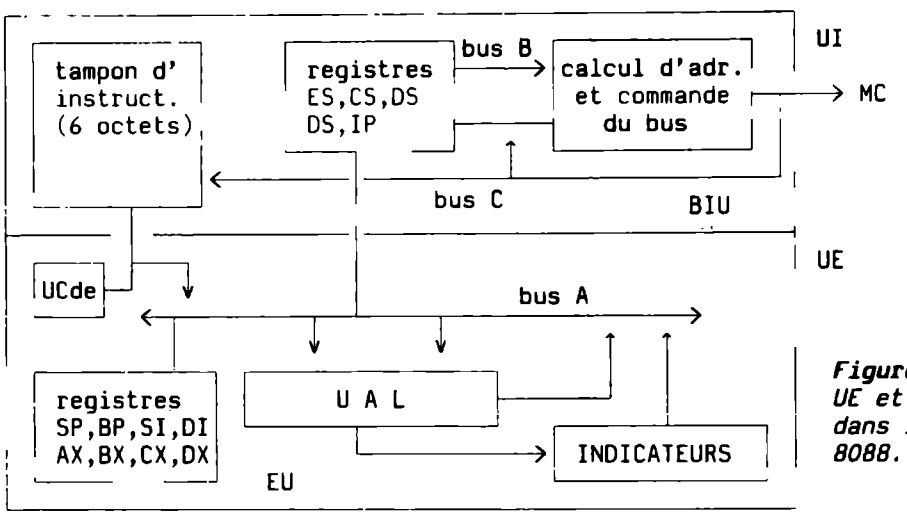


Figure 17f: UE et UI dans le 8088.

Si dans les opérateurs parallélisme et anticipation peuvent cohabiter, il en va de même dans le processeur où l'on trouvera des circuits dupliqués, comme l'UI dans les AS9000 déjà cités (figure 17g).

Dans les DPS88 l'UI et l'UE (il y a en fait 3 UE) sont toutes deux pipelinées en 5 étapes indiquées ci-dessous:

UNITE D'INSTRUCTION	UNITE D'EXECUTION
. incrémentation de CO	. décodage de l'instruction
. calcul de l'adresse virtuelle de la paire d'instructions	. calcul de l'adresse virtuelle de l'opérande
. extraction du numéro de page de l'antémémoire	. extraction du numéro de page de l'antémémoire
. lecture de l'antémémoire	. lecture de l'antémémoire
. envoi de la paire d'instructions dans le tampon d'instructions	. exécution de l'instruction ou transfert à un opérateur spécialisé

Pour les connaisseurs du 3083 d'IBM, notre UI correspond à l'UE plus le BCE, notre UE au VFE (calculs décimaux) plus l'EE (binaire).

17.6 – Multiprocesseurs.

Le vocabulaire ANSI définit les multiprocesseurs comme des systèmes composés d'au moins deux unités de traitement avec une commande

intégrée. [ENS77] donne comme caractéristique des multiprocesseurs:
- un ou plusieurs processeurs avec des fonctionnalités comparables,
- tous les processeurs ont accès à une mémoire centrale appelée **mémoire commune**; ceci élimine les machines de type TANDEM, STRATUS,
- tous les processeurs ont accès aux canaux, unités de commande (contrôleurs), périphériques; ceci élimine certains multiprocesseurs comme les AP (Attached Processor) d'IBM, les VAX Clusters,
- le système est piloté par une seule copie du système d'exploitation.

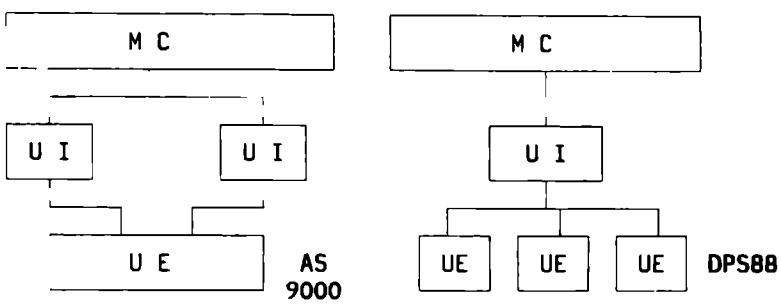


Figure 17g: exemples de processeurs utilisant le parallélisme au niveau du traitement des instructions.

Dans cette section, nous considérerons comme multiprocesseurs les ordinateurs n'ayant en mémoire centrale qu'une seule copie du système d'exploitation, et rangerons les autres types d'ordinateurs dans la catégorie des multisystèmes.

Certains ordinateurs sont conçus comme des multiprocesseurs dès leur conception initiale (DPS8, 1100 de Sperry, entre autres), d'autres le deviennent lorsque les conditions de performance (haut de gamme plafonnant), de mode, l'imposent (cas du VAX avec le VAX-11/782, du 850 de PRIME).

Ces multiprocesseurs correspondent aux MIMD: flot multiple d'instructions, flot multiple de données selon la classification de Flynn. Si les structures UIMD conviennent bien au calcul scientifique puisqu'elles dupliquent la ressource de calcul (l'UT), tous les algorithmes ne se prêtent pas efficacement à la "vectorisation"; [AUG84] cite des gains de 10 à 20 % seulement par rapport à un ordinateur scalaire lorsque l'on effectue des calculs de triangulation sur CRAY-1. Notons encore en ce qui concerne le vocabulaire que les UIMD sont parfois qualifiés de "synchrones" alors que les MIMD le sont de "asynchrones".

Il est intéressant de noter qu'IBM est venu relativement tard aux multiprocesseurs, avec les 3081 en fait, après avoir eu quelques modèles bi-processeurs. IBM se limite actuellement aux bi- et quadriprocesseurs.

On distingue généralement trois types de structure, en fonction de l'interconnexion des processeurs et des mémoires:
- les **matrices de connexion**,
- les **mémoires multiportes**,
- les **bus**.

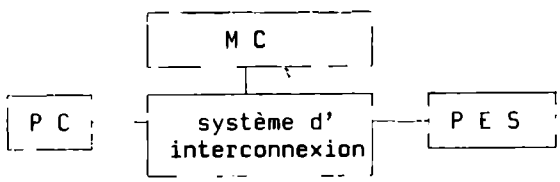


Figure 17h: structure générale des multiprocesseurs.

PC : processeur central
MC : mémoire centrale
PES: processeur d'entrée sortie

17.6.1 – Matrices de connexion.

Comme dans les centraux téléphoniques à commutation spatiale, une matrice de commutation permet de mettre en relation tout processeur avec tout bloc mémoire.

Cette solution présente un coût élevé, mais par contre offre un système sans blocage, dans la mesure où les blocs de mémoire sont capables de traiter plusieurs requêtes simultanément.

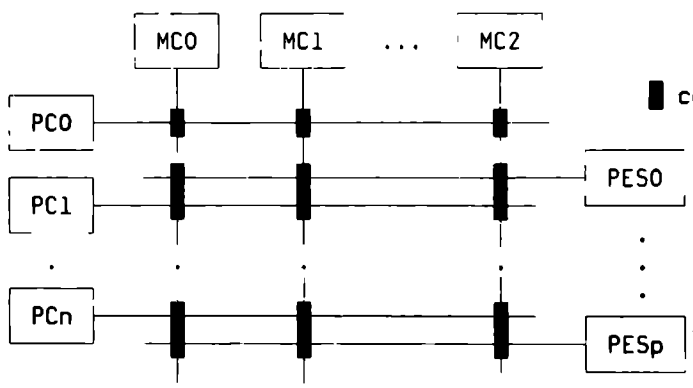


Figure 17i: matrice de connexion.

MC : mémoire centrale
PC : processeur central
PES: processeur d'entrée-sortie

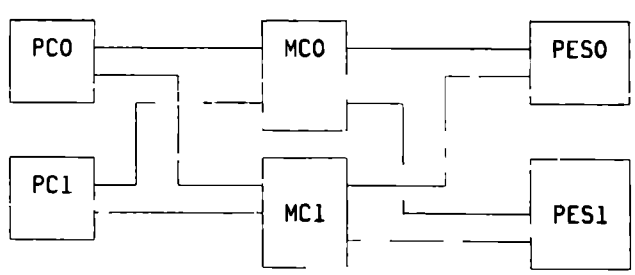


Figure 17j: mémoires multiportes.

17.6.2 – Mémoires multiportes.

C'est le cas des DPS8, Sperry 1100, VAX-11/782. La mémoire constitue le centre du sous-système central et chaque bloc mémoire, ou plus généralement un contrôleur mémoire et les blocs qu'il contrôle (cf. chapitre 18), offre un accès à chacun des processeurs, huit au total dans les DPS8 (huit processeurs centraux ou d'entrée/sortie).

Cette structure nécessite autant de points de connexion que les matrices de connexion (crossbar), mais les éléments de commutation sont répartis dans les blocs mémoire.

17.6.3 – Bus.

La structure en bus commun multiplexé temporellement peut être vue comme une simplification de la matrice de connexion.

Le bus devient vite un élément de blocage dès que le trafic augmente et ces structures sont réservées à des machines de puissance moyenne ou à faible trafic entre les éléments interconnectés.

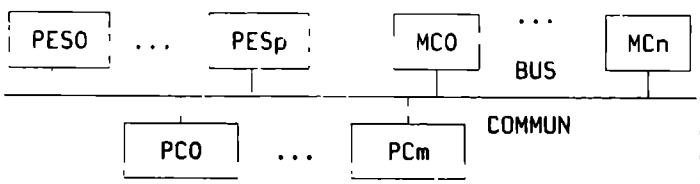


Figure 17k: réseau d'interconnexion assuré par un bus.

17.6.4 – Symétrie et asymétrie des processeurs.

Nous nous intéressons dans cette section à la façon dont les processeurs centraux sont constitués, à la manière dont ils sont reliés au reste du système, et aux rôles qu'ils y jouent.

Tous les PC ne sont pas toujours reliés au reste du système de façon identique. C'est par exemple le cas des configurations AP (Attached Processor) chez IBM avec les 158AP, 3033AP, etc.

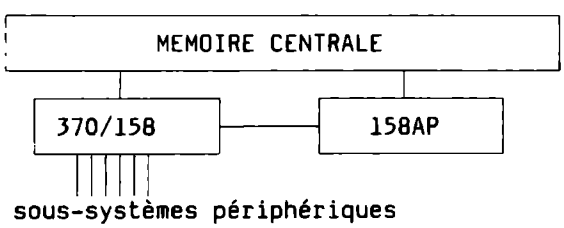


Figure 17l: configuration AP chez IBM.

Comme le montre la figure 17l, seul un des processeurs, appelé "maître", accède aux sous-systèmes périphériques. De plus c'est lui qui distribue (dispatch) le travail au processeur "esclave", d'où une moindre efficacité de l'ensemble. Dans certaines configurations AP (3033AP par exemple) le processeur attaché peut lancer des entrées-sorties.

Dans les configurations dites MP (MultiProcessor) chez IBM chaque processeur peut accéder aux sous-systèmes périphériques, mais aux siens seulement (figure 17m).

Dans les 158MP chaque processeur dispose de sa propre mémoire centrale, bien qu'il n'y ait qu'une seule copie du système d'exploitation. Si l'un des processeurs tombe en panne, la mémoire qui lui appartient est inutilisable.

Ces asymétries de **construction** créent donc de mauvaises conditions pour une bonne gestion de l'ensemble des ressources, ce que veut atteindre un système d'exploitation unique. Les structures qui offrent ces conditions de bonne gestion globale sont celles que nous appellerons **banalisées** (cf figure 17n). Pyramid utilise le terme "d'isoprocesseur" pour cette banalisation des deux processeurs de son 98X.

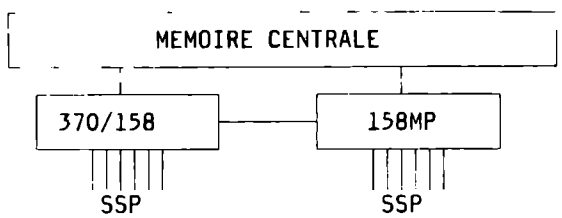


Figure 17m: configuration MP chez IBM.

On y constate que toutes les ressources sont accessibles depuis tout processeur. La banalisation de construction peut cependant être remise en cause par le logiciel. Ainsi sur les DPS8 gérés par GCOS3, le processeur zéro est le seul à traiter les interruptions. Il en est de même dans les biprocesseurs des séries 308X d'IBM appelés **dyadiques** où l'un des processeurs a un rôle privilégié. Sur le PRIME 850 un seul des deux processeurs peut lancer des entrées/sorties. Sur les 3200MPS de Perkin Elmer un des onze processeurs joue un rôle privilégié. Dans le VAX-11/782 où l'un des processeurs est appelé "attaché", seul le "primaire" peut distribuer le travail; l'attaché ne peut pas dérouler du code en mode privilégié, tout comme dans les premières configurations IBM à deux processeurs; il ne peut pas être interrompu pour prendre un travail plus prioritaire (technique de pré-emption que nous étudierons dans le Tome II).

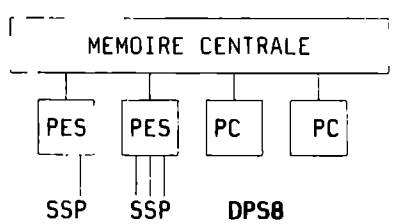
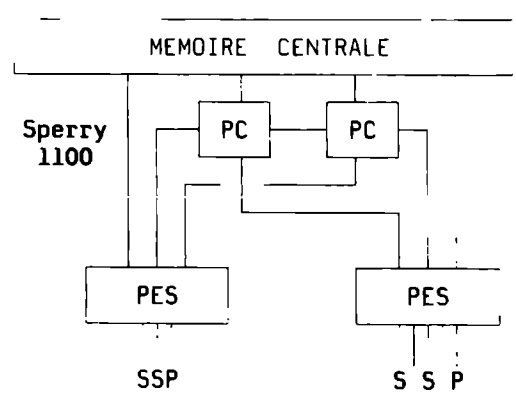


Figure 17n: multiprocesseurs symétriques (banalisés) par construction.



Jusqu'ici nous avons implicitement supposé une identité des architectures externe et interne des différents processeurs. Il n'en est pas ainsi dans certaines configurations qui contiennent des processeurs **asymétriques**. C'est par exemple le cas chez CDC et CRAY:

- un ou plusieurs (deux chez CDC) processeurs centraux déroulent les programmes des utilisateurs, c'est-à-dire des calculs scientifiques intensifs,
- jusqu'à vingt processeurs périphériques (PP) déroulent les instructions du système d'exploitation. Ce sont des mini-ordinateurs.

Dans le cas de CDC l'évolution a conduit à implanter une partie croissante du système d'exploitation dans les processeurs centraux.

Une structure analogue de **multiprocesseurs hétérogènes** se retrouve dans les minis 32 bits de Norsk Data où un mini 16 bits (de la série ND10 ou ND100) gère le système.

Citons encore certains modèles de NCR dans lesquels les processeurs sont de même architecture externe mais d'architecture interne différente afin de fournir des modèles de puissances diverses.

Nous mentionnerons enfin le cas des ordinateurs individuels comme le GOUPIL 3 de SMT conçu autour d'un microprocesseur 6809 supportant le système d'exploitation FLEX9. Etant donné l'importance du système d'exploitation CP/M dans le domaine des huit bits, SMT offre une option CP/M qui conduit à ajouter un second processeur, un Z-80, pour que le Goupil fonctionne sous CP/M. Mais à un moment donné, un seul des systèmes d'exploitation est actif, un seul processeur est donc en fonctionnement. Des situations analogues, avec des degrés d'intégration des systèmes d'exploitation plus ou moins élevés, se retrouvent sur un certain nombre de micro-ordinateurs qui n'avaient pas choisi initialement CP/M, et surtout MS-DOS.

17.7 – Les multisystèmes.

Les multisystèmes, composés généralement de plusieurs ordinateurs et non pas de plusieurs processeurs uniquement, correspondent à une très grande variété de situations.

Un critère de classement est le degré de coopération entre les différents systèmes d'exploitation qui habillent ces ordinateurs.

Dans certains cas cette coopération est de très bas niveau puisque l'un des ordinateurs se comporte comme un périphérique de l'autre ("émulation de terminaux lourds"), dans d'autres les systèmes d'exploitation dialoguent d'égal à égal et l'on parlera de "systèmes d'exploitation distribués".

On le voit, le problème est surtout un problème de logiciel et il sera donc étudié dans le Tome II. Le cas de quelques systèmes sera cependant abordé dans ce tome au cours du chapitre sur les machines "tolérant les pannes" (fault tolerant systems) puisque nombre d'entre elles sont basées sur des solutions matérielles, au niveau du sous-système central en particulier.

17.8 – Remarques sur les classifications.

La classification des ordinateurs n'est ni chose aisée, ni lorsqu'elle existe, résultat reconnu et accepté par tous:

- celle de Flynn définit quatre classes selon que les flots de données et/ou d'instructions sont simples ou multiples, le monoprocesseur

classique entrant dans la catégorie UIUD (flot unique d'instructions, flot unique de données);
- celle de Shore (1973) définit six classes en utilisant comme critère l'organisation des éléments constitutifs du sous-système central:
  * l'unité de commande,
  * l'unité de traitement,
  * la mémoire des instructions,
  * la mémoire des données.
- celle de [COM83] porte sur les supercalculateurs.

chapitre 18

parallélisme et anticipation au niveau des mémoires

Nous avons vu que le rôle de la mémoire centrale était celui d'un adaptateur, d'un accélérateur de vitesse, entre le processeur central, qui assure la fonction de **traitement**, et les mémoires périphériques qui assurent celle de **stockage**. Tout étant possible, il existe des micro-ordinateurs où la mémoire centrale joue un rôle de stockage (cas de l'Olivetti M/10); ce phénomène est récent et ne concerne que des produits de bas de gamme.

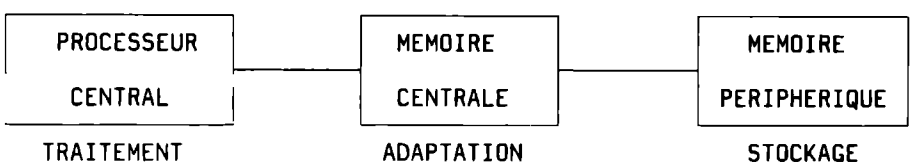


Figure 18a: fonctions des différents composants.

La mémoire centrale peut aussi constituer un frein pour le processeur, d'où les registres dont est doté ce dernier. Les registres jouent vis-à-vis de la mémoire centrale, le rôle que cette mémoire centrale joue vis-à-vis des mémoires périphériques.

Cette **hiérarchie de mémoires** relève de l'anticipation puisque la fonction de chargement des informations entre mémoires périphériques (MP) et centrale (MC) est décomposée en deux sous-fonctions:

- chargement depuis les mémoires périphériques dans la mémoire centrale,
- puis chargement de la mémoire centrale dans le processeur.

En faisant intervenir les registres, nous ajouterions une troisième sous-fonction.

Les techniques d'anticipation et de parallélisme, **transparentes** pour l'utilisateur au niveau de l'architecture externe (les registres et la mémoire centrale ne le sont bien évidemment pas), feront intervenir:

- les blocs de mémoire au niveau du parallélisme,
- les antémémoires et les tampons d'instruction ou de données en ce qui concerne l'anticipation.

Le nombre d'étages dans la hiérarchie des mémoires va donc augmenter de façon conséquente, et ce n'est pas fini puisque nous en trouverons d'autres dans les sous-systèmes périphériques (Tome IV).

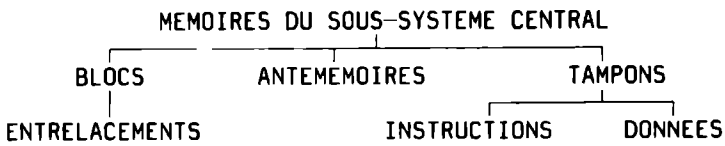


Figure 18b: techniques de P&A dans les mémoires.

18.1 – Blocs de mémoire.

La mémoire est une ressource fortement sollicitée par le processeur, et elle l'est d'autant plus que ce dernier est pipeliné, et que l'on a affaire à un multiprocesseur.

La mémoire ayant un temps de cycle et un temps d'accès donnés, son découpage en plusieurs blocs indépendants va lui permettre de satisfaire plusieurs requêtes simultanément.

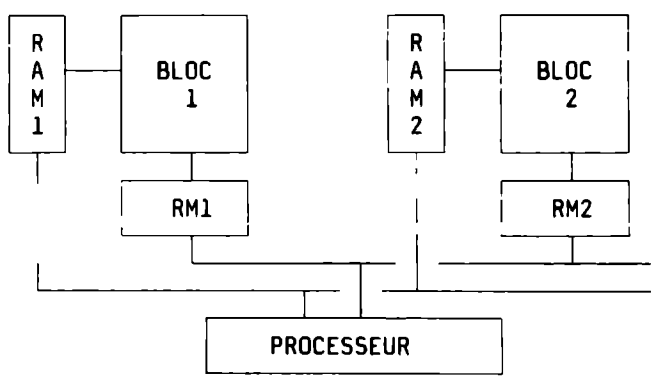


Figure 18c: mémoire centrale décomposée en deux blocs.

RM : registre tampon mémoire
RAM: registre adresse mémoire

On constate sur le schéma de la figure 18d que le processeur n'a pas à attendre la fin d'une requête vers un bloc pour lancer une nouvelle requête vers le second bloc, ce qu'il devrait faire si la mémoire était monolithique. Soit t le temps pendant lequel est occupé le bus (t=t2-t1) pour transférer une information entre le processeur et les registres RMi ou RAMi. Ce transfert entre registres est très rapide, plus court que les temps d'accès et de cycle des blocs de mémoire. Soit T le temps de cycle de la mémoire et Ta son temps d'accès. Avec un seul bloc mémoire, les deux accès dureraient T+Ta, alors qu'avec les deux blocs, ils durent Ta+t, d'où un gain important puisque le processeur n'a pas à attendre de façon inactive. Cette structure de blocs entrelacés apparaît sous les termes de BSE (basic storage element) et BSM dans les IBM 3083.

Si le nombre de blocs augmente, il est préférable de dupliquer les bus afin d'offrir plusieurs chemins indépendants vers les blocs de mémoire.

Certains constructeurs donnent des **temps d'accès apparents**. Supposons qu'une mémoire de temps d'accès de 500 ns soit découpée en quatre blocs indépendants. Toute lecture demande 500 ns d'attente, mais les constructeurs parleront d'un temps d'accès de 500/4 soit 125 ns, en oubliant parfois d'ajouter apparent. Ils veulent ainsi traduire le fait que si les accès sont bien répartis, la mémoire aura un débit (bandwith) de 8 000 000 de mots par seconde, comme une mémoire monolithique de temps de cycle égal à 125 ns.

18.2 – Entrelaccment des adresses.

La situation présentée ci-dessus est quelque peu idyllique, dans la mesure où l'on suppose que les requêtes du processeur se distribuent harmonieusement entre les blocs. Il n'en est pas toujours ainsi.

Une contribution à cette bonne distribution peut se faire au niveau des programmes, en séparant par exemple le code et les données. On trouve ceci dans les 1100 de Sperry, ordinateur avec registres de base et limite [RB,RL] protégés comme nous l'avons vu dans le chapitre sur les modes d'adressage. Il a en fait deux couples [RB,RL], l'un pour les données, l'autre pour les instructions [D-bank et I-bank]. En disposant par exemple le code en début de mémoire et les données en bas de cette mémoire on aura une bonne répartition automatique des accès.

La séparation du code et des données apporte aussi d'autres avantages comme nous le verrons dans le Tome V lorsque nous aborderons les sous-programmes réentrants.

Mais il y a une solution purement matérielle pour améliorer la distribution des adresses, c'est ce que l'on appelle **l'entrelacement des adresses.**

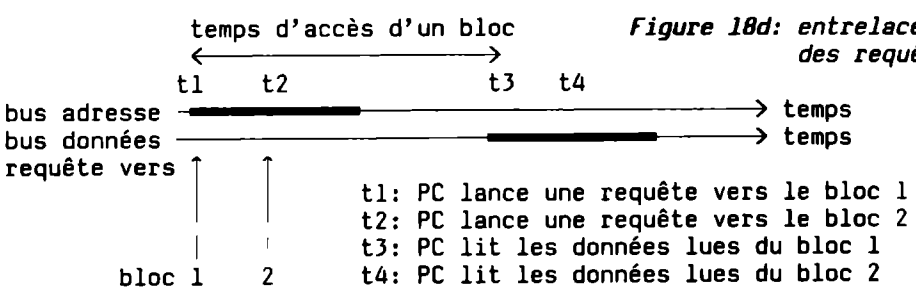


Figure 18d: entrelacement des requêtes.

Les différents blocs peuvent de plus être **déphasés**: leurs cycles d'accès sont décalés d'un cycle de base (cycle du PC). C'est le cas dans certains superordinateurs afin d'alimenter les opérateurs pipeline de façon continue.

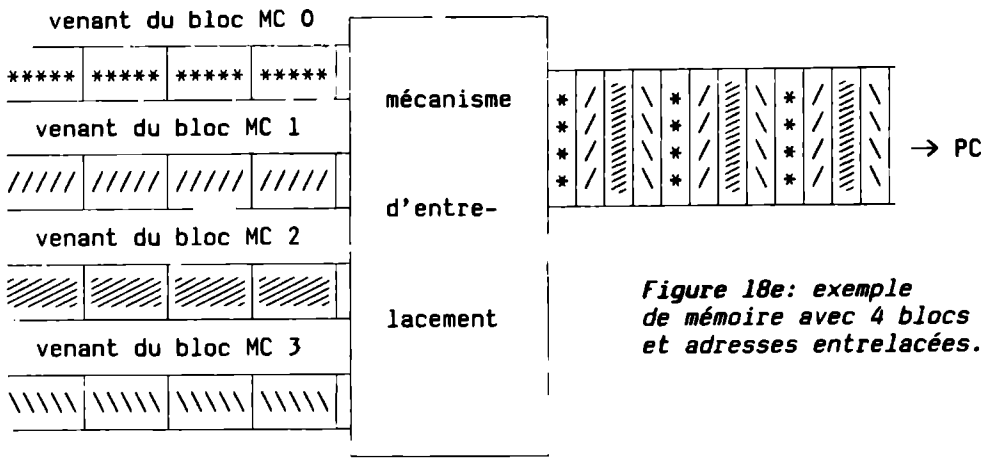


Figure 18e: exemple de mémoire avec 4 blocs et adresses entrelacées.

L'entrelacement des adresses complique la gestion de la mémoire centrale par le système d'exploitation: il suffit d'imaginer le cas où un des blocs a un morceau de mémoire en panne.

NOMBRE DE NIVEAUX	PROCESSEUR
4	370/168-3
8	3033
2 ou 4	CYBER 825
2 ou 4	66 de Bull
16	CRAY-1 *
32	CRAY XMP-2 **

Tableau 18f: exemples de nombres de blocs avec entrelacement (interlace).

* 1 chemin d'accès vers la mémoire par processeur.
** 4 chemins par processeur: 2 en écriture, 1 en lecture, 1 pour les entrées/sorties.

18.3 – Objectif des antémémoires.

L'antémémoire est un dispositif **transparent** pour l'utilisateur normal, c'est-à-dire l'utilisateur ayant à résoudre des problèmes applicatifs et non à gérer le processeur.

L'antémémoire se trouve à l'intérieur du processeur, entre les registres et la mémoire centrale, étage supplémentaire dans la hiérarchie des mémoires.

Comme de nombreuses notions développées dans le cadre de ce tome (l'adressage relatif par exemple) l'antémémoire est basée sur la propriété de **localité** que possède le déroulement des instructions et l'accès aux données. Quand le processeur exécute une instruction, il y a de fortes chances (couramment 75 %, davantage en scientifique) pour qu'il exécute l'instruction qui la suit dans la mémoire centrale (localité spatiale). De même lorsqu'une instruction référence une donnée en mémoire centrale, il y a de fortes chances pour qu'elle référence ultérieurement les données qui lui sont contiguës, surtout si l'on traite des tableaux.

L'idée qui conduit aux antémémoires (cache memory en américain, high speed buffer chez IBM entre autres) est qu'il faut amener aussi vite que possible à l'intérieur du processeur les instructions qui suivent celle en cours d'exécution, et d'amener aussi les données placées près de celles que l'instruction est en train de traiter. C'est l'antémémoire qui contiendra ces données lues de façon anticipée.

Avant de présenter les antémémoires, nous en étudierons une forme simplifiée que l'on appelle les **tampons d'instructions**. Il peut aussi y avoir des **tampons de données**, mais moins fréquemment.

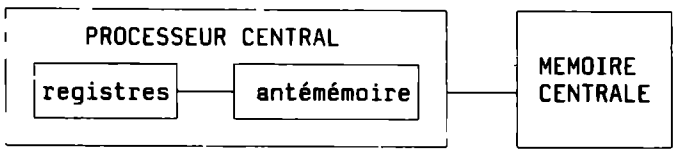


Figure 18g: place de l'antémémoire.

18.4 – Tampons d'instructions.

Un tampon d'instructions (instruction buffer, instruction stack en américain) est une petite mémoire (cf. les tailles données dans le tableau 18h) dans laquelle le processeur, sa partie unité d'instruction UI plus exactement, accumule des instructions avant de les traiter.

Comme on peut le constater cette technique est ancienne (la LARC d'UNIVAC, ancien nom de Sperry, en disposait en 1959) et très répandue puisqu'on la trouve dans les microprocesseurs aussi bien que dans les calculateurs scientifiques de haut de gamme.

Ces tampons sont efficaces tant que l'exécution se déroule en séquence puisque c'est ainsi que les instructions sont chargées de façon anticipée dans le tampon. En cas d'instruction de branchement inconditionnel, ou si le branchement conditionnel se fait dans le "mauvais" sens, les instructions pré-chargées sont inutiles et le tampon est remis à zéro: le processeur perd du temps.

Pour surmonter ce problème lié aux ruptures de séquence, les architectes ont dupliqué les tampons d'instructions: le tableau 18h montre que le 168-3 en a deux, le 3033 en dispose de trois. Dès qu'une instruction de branchement est chargée dans un tampon, une des deux branches est rangée dans ce tampon, et l'autre branche dans le second tampon. Ainsi, en cas de branchement, les deux séquences possibles sont déjà dans le processeur.

Il existe bien sûr des branchements (indexés par exemple) pour lesquels il est difficile de prévoir longtemps à l'avance les adresses des séquences possibles. Dans le cas d'un branchement dépendant du contenu d'un registre il faut attendre que ce contenu ait été défini. [MDO84] donne dans le cas de systèmes 370 et de mix COBOL, des chiffres exprimant la dépendance due à la génération d'adresse:

- 12 % des instructions sont ainsi touchées, c'est-à-dire que leur exécution dépend d'une adresse générée par des instructions qui précédent et dont elles doivent attendre l'exécution;
- 88 % de ces dépendances sont créées par les instructions de chargement L ou LR;

- deux tiers de ces chargements servent à charger des registres pour calculer des adresses de base (il n'y a pas d'indirection dans les 370) ou le contenu de registres d'index;
- un cas très fréquent est constitué par la séquence L-BCR utilisée pour réaliser des boucles.

NOMBRE	TAILLE	PROCESSEUR
1	12 mots	CYBER de haut de gamme
1	8 mots	CDC 6600 en 1965
1	8 octets	VAX-11/780
2	16 octets	370/168-3
3	32 octets	3033 IBM
4	64 parcelles	CRAY-1
4	128 parcelles	CRAY XMP-2
1	6 octets	8086 INTEL
1	8 octets	NS 16032
1	7 instruct	HARRIS 800
	32 mots	DPS88

Figure 18h: exemples de tampons d'instructions.

[MDO84] décompose comme suit le temps de traitement I d'une instruction:

$$I = E + D + S$$

avec
- E le temps moyen d'exécution de l'instruction lorsque le pipeline est toujours plein;
- D est le temps moyen perdu par instruction du fait des "trous" du pipeline: temps lié aux conflits de chemin, de dépendance des registres, etc.,
- S est le temps moyen par instruction passé à accéder à la mémoire, retard provoqué par une instruction ou un opérande qui n'est pas dans le tampon et doit donc être cherché en mémoire centrale.

Pour limiter ces temps morts, le Spectrum de HP emploie une technique de branchements "différés" (cf. [BIR85]) qui consiste à écrire le branchement avant la dernière instruction de la séquence, au lieu de l'écrire après comme on le fait habituellement. Dans le Ridge 32 les instructions de branchement comportent un bit de "prédiction" afin de faciliter le travail du processeur; s'il est égal à un le processeur pré-charge les instructions qui se trouvent à l'adresse de branchement, s'il est égal à zéro, il pré-charge l'instruction suivante; ce bit est positionné par les programmeurs ou les compilateurs.

18.4.1 – Tampons de micro-instructions.

Ce qui est fait au niveau du processeur peut bien évidemment être fait au niveau de la **micro-machine** que constitue l'unité de commande microprogrammée. Et pour les mêmes raisons. S'il existe des mémoires de commande avec plusieurs blocs et avec adresses entrelacées, il y en a aussi qui disposent de tampons de micro-instructions. C'est par

exemple le cas des VAX-11/750 et 780 qui ont un tampon (80 bits et 96 respectivement) leur pemettant de stocker la micro-instruction suivante pendant l'exécution de la courante.

18.5 – Fonctionnement de l'antémémoire.

Revenons maintenant au cas général de l'antémémoire après avoir noté (figure 18i) qu'elle peut coexister avec les tampons d'instruction. Chaque fois que le processeur va lire une case en mémoire centrale, deux cas vont se présenter:

- soit le contenu de cette case se trouve déjà dans l'antémémoire et la demande s'arrête au niveau de l'antémémoire. L'accès est donc plus rapide (une antémémoire est couramment dix fois plus rapide que la mémoire centrale) et il ne charge pas la mémoire centrale, la laissant disponible pour d'autres processeurs dans le cas d'un multiprocesseur;
- soit le contenu de cette case ne se trouve pas dans l'antémémoire. Il faut alors aller dans la mémoire centrale et cette fois le processeur va être sensible à la vitesse de la mémoire centrale. Mais le transfert entre la mémoire centrale et le processeur ne va pas se limiter à la case demandée, il va concerner tout un groupe de cases que l'on appellera **bloc** et dans lequel se trouve la case initialement concernée. On voit donc apparaître le mécanisme **d'anticipation** annoncé plus haut. C'est ainsi que l'antémémoire va peu à peu se remplir et accumuler des informations qui vont être demandées par le processeur plus tard.

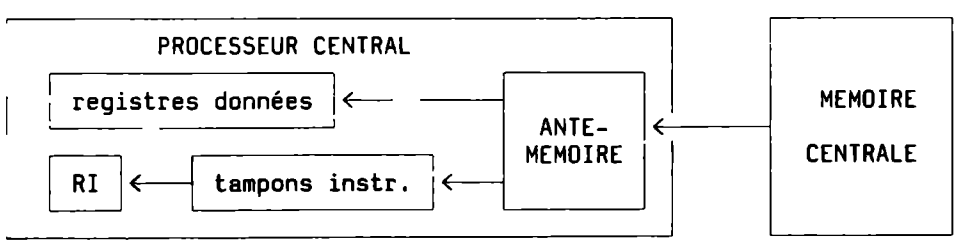


Figure 18i: place des tampons d'instruction et de l'antémémoire.

Le fonctionnement de base de l'antémémoire ainsi défini, nous allons en détailler les différents aspects. Notons que cette antémémoire ne fonctionne **qu'en lecture**: lorsque le processeur écrit en mémoire centrale, il subit le freinage du à la relative lenteur de cette mémoire. Nous étudierons plus loin des dispositifs qui permettent de masquer en partie ce ralentissement par l'écriture. Il faut cependant noter que les lectures sont plus fréquentes que les écritures; [FRE85] avance 85 % de lectures contre 15 % d'écritures dans les applications courantes.

18.6 – Efficacité d'une antémémoire.

L'efficacité d'une antémémoire se mesure au pourcentage de lectures qu'elle satisfait, sans solliciter la mémoire centrale lorsque le processeur émet sa requête:

$$\frac{100 * \text{nombre de lectures en antémémoire}}{\text{nombre total de lectures faites par le processeur}}$$

Ce nombre appelé aussi **taux de réussite** ou taux de succès (hit ratio en américain) est fréquemment annoncé comme égal à 95 % par les constructeurs. Ceci reflète plus ou moins la réalité car, comme nous le verrons par la suite, l'efficacité d'une antémémoire (AM) peut dépendre des types de programme que l'on exécute. Or les constructeurs ont du mal à trouver des programmes représentatifs de ce que font les utilisateurs lorsqu'ils conçoivent, mesurent, modélisent leurs antémémoires. Sur certains ordinateurs annoncés pour un taux de réussite de 90 %, on pourra trouver des environnements où ce taux descend à 50 !

Pour illustrer ce taux de succès prenons l'exemple d'une mémoire avec un temps d'accès de 1 000 ns et d'une antémémoire dont le temps d'accès est de 100 ns. Si t est le taux de succès, le **temps d'accès apparent**, celui observé en moyenne par le processeur, sera égal à

$$100 * t + 1000 * (1-t)$$

soit 190 ns pour un taux de succès de 0,9 (90 %).

Dans le cas de multiprocesseurs disposant chacun d'une antémémoire, on parlera de taux de succès **latéral** pour caractériser le nombre de requêtes satisfaites par les antémémoires des autres processeurs.

18.7 – Structure d'une antémémoire.

L'antémémoire contient non seulement des informations que le processeur va traiter ou a traité, mais toute la logistique nécessaire à gérer ces informations: répertoire, circuits, etc.

18.7.1 – Blocs, colonnes et niveaux.

Une antémémoire contient un certain nombre de **blocs** de la mémoire centrale (le tableau 18m donne quelques exemples de taille de bloc). Notons que l'antémémoire conduit à une **duplication** physique de l'information, et contrairement à celle produite par les registres, cette duplication n'est pas gérée par l'utilisateur, mais par le matériel lui-même, de façon **transparente** au niveau de l'architecture externe, il est utile d'insister sur ce point.

Soit **B** le nombre de blocs contenus dans la mémoire centrale et **b** celui dont dispose l'antémémoire. Quand un bloc va arriver dans l'antémémoire, où va-t-on le ranger? Une réponse simple conduit à utiliser le premier bloc libre trouvé dans l'antémémoire. En fait cette solution n'est jamais retenue car elle conduit à des circuits trop complexes, donc trop chers, et parfois trop lents: dans certaines technologies les mémoires deviennent très vite trop lentes quand leur taille augmente; on préfère disposer de plusieurs mémoires plus petites.

COLONNE 0	COLONNE 1	COLONNE 2	COLONNE 3
bloc a	bloc b	bloc c	bloc d
bloc e	bloc f	bloc g	bloc h

ANTEMEMOIRE

COLONNE 0	COLONNE 1	COLONNE 2	COLONNE 3
bloc 0	bloc 1	bloc 2	bloc 3
bloc 4	bloc 5	bloc 6	bloc 7
bloc 8	bloc 9	bloc 10	bloc 11
bloc 12	bloc 13	bloc 14	bloc 15
bloc 16	bloc 17	bloc 18	bloc 19

MEMOIRE CENTRALE

Figure 18j: cas d'une antémémoire à 4 colonnes et 2 niveaux.

Dans le cas des antémémoires les blocs (block en américain) sont regroupés en **colonnes** (column, set en américain). Soit **C** le nombre de colonnes de l'antémémoire. La mémoire centrale sera elle aussi vue comme un ensemble de blocs regroupés en C colonnes. Quand un bloc arrive de la mémoire centrale il va obligatoirement dans un bloc de l'antémémoire qui a le même numéro de colonne. Ceci réduit le nombre de choix pour placer le bloc en antémémoire et donc simplifie les circuits la composant. Ce nombre de choix est appelé le nombre de **niveaux** (level en américain) de l'antémémoire.

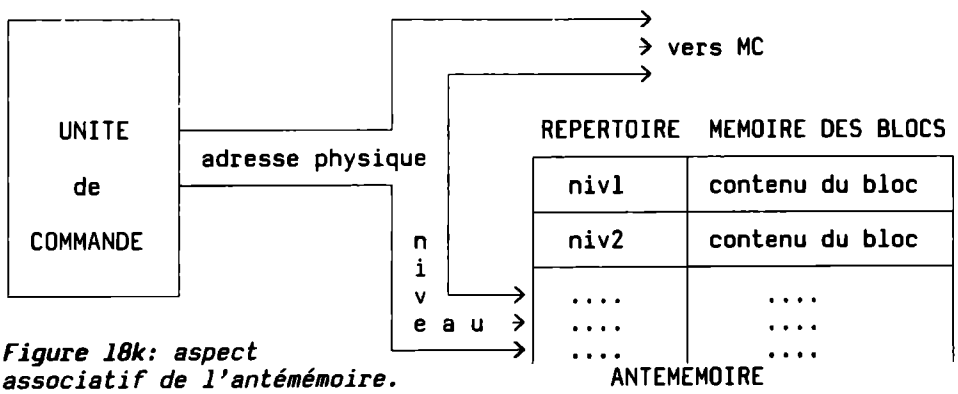


Figure 18k: aspect associatif de l'antémémoire.

Si C est égal à un (il n'y a pas de notion de colonne en fait) on parlera d'antémémoire **complètement associative** (fully associative cache memory en américain), le terme d'associatif étant expliqué plus loin.

Soit **N** le nombre de niveaux dans la mémoire centrale et n celui

de l'antémémoire. Si n est égal à 1, cas de l'antémémoire du VAX-11/750 par exemple, il n'y a aucun choix de placement et on parlera d'antémémoire "directement couverte" (directly mapped cache memory en américain).

Dans l'exemple de la figure 18j, les blocs mémoire 0, 4, 8, 12, 16, etc. pourront aller dans l'un des deux blocs a et e de l'antémémoire.

18.7.2 – Le répertoire.

Nous venons de voir la structure de la partie mémoire qui contient les blocs d'information. Elle est généralement en technologie plus rapide que la mémoire centrale: bipolaire contre MOS pour cette dernière.

Mais l'antémémoire doit aussi stocker l'adresse en mémoire centrale de l'information qu'elle contient, sinon elle ne pourra pas répondre aux requêtes de l'unité de commande. Le découpage en colonnes permet de réduire la taille de cette adresse puisque les numéros de colonne sont les mêmes en mémoire centrale et dans l'antémémoire.

Si nous considérons l'adresse physique de la figure 18l, nous voyons que seul le numéro de niveau dans la mémoire centrale a besoin d'être stocké dans l'antémémoire. Ces numéros de niveau sont stockés dans une seconde mémoire de l'antémémoire que l'on appelle le **répertoire** (catalogue, catalog, directory en américain).

numéro de niveau (ou de "ligne")	numéro de colonne	déplacement dans le bloc

Figure 18l: décomposition d'une adresse physique lorsque l'on veut faire apparaître la structure de l'antémémoire.

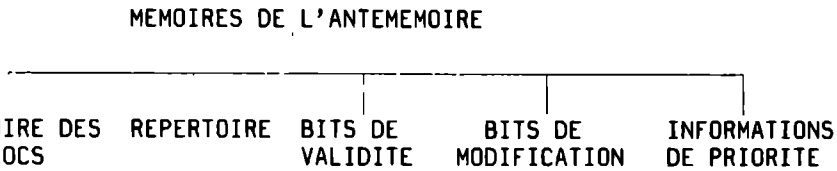
C'est lui qui donne à l'antémémoire cette caractéristique **d'associativité**. D'une façon générale une mémoire associative est une mémoire à laquelle on n'accède pas par adresse comme dans le cas de la mémoire centrale, mais par une partie de son **contenu**. Quand l'unité de commande envoie une requête à la mémoire centrale, elle envoie une adresse: quand l'antémémoire intercepte cette adresse, elle va extraire la partie numéro de niveau et la comparer avec tous les numéros qu'elle possède dans son répertoire. Ce n'est donc pas une recherche par adresse, mais une recherche par contenu qu'elle réalise.

Ce répertoire peut être doublé (cas des DPS88) pour des raisons de performance: l'un est utilisé par l'unité de commande du processeur, l'autre par les autres processeurs centraux ainsi que par les processeurs d'entrée/sortie.

18.7.3 – Autres mémoires de l'antémémoire.

Nous avons déjà présenté la mémoire interne (mémoire des blocs) et le répertoire de l'antémémoire. Pour sa gestion interne l'antémé-

moire dispose généralement d'autres mémoires:
- les bits de **validité** qui indiquent si une entrée (numéro du niveau du bloc en mémoire centrale et contenu de ce bloc) de l'AM est occupée ou non;
- les bits de **modification** qui indiquent si le contenu du bloc a été modifié dans l'antémémoire depuis qu'il a été chargé à partir de la mémoire centrale. Cette information est indispensable lorsque l'on ne met pas automatiquement à jour la mémoire centrale, technique que nous étudierons dans les algorithmes de vidage;
- les informations de **priorité** lorsque l'antémémoire dispose d'un mécanisme cherchant à conserver les blocs les plus utilisés. Nous verrons ce point dans le cadre des algorithmes de placement.



18.8 – Gestion de l'antémémoire.

La gestion de l'antémémoire fait appel à plusieurs algorithmes que nous ne ferons que présenter. Leur développement serait trop volumineux. Ces algorithmes sont au nombre de trois avec quelques variantes internes:
- l'algorithme de **remplissage** (catch fetch algorithm) est celui qui décide **QUAND** il faut amener l'information dans l'antémémoire;
- les algorithmes de **placement** et **remplacement** décident **OU** doit être rangée l'information dans l'antémémoire;
- l'algorithme de **vidage** régit la **MISE A JOUR** de la mémoire centrale.

18.8.1 – Algorithme de remplissage.

Il comprend deux variantes selon que l'on remplit l'antémémoire:

- sur **demande**, c'est-à-dire lorsque l'unité de commande la requiert,
- de façon **anticipée**. On parle alors de **pré-remplissage**. L'information est amenée avant qu'elle ne soit demandée par l'unité de commande. L'algorithme de pré-remplissage a trois grandes décisions à prendre:
  * quand lancer un pré-remplissage,
  * quels blocs pré-remplir,
  * quel état donner aux blocs pré-chargés.

  Le 470V/8 par exemple a un algorithme de pré-chargement basé sur les échecs. Le critère de pré-remplissage peut aussi être l'écriture: on remplit sur écriture (fetch on write).

Notons que certaines informations peuvent ne pas être chargées en antémémoire dans le cas de multiprocesseurs avec des antémémoires peu complexes (cas des segments partageables de MULTICS sur les DPS8M).

Lorsqu'un échec (miss) se produit, c'est-à-dire lorsque l'antémémoire ne contient pas l'information demandée il y a deux grands types

de réaction:
- il y a d'abord remplissage de l'antémémoire depuis la mémoire centrale, puis relance de la lecture qui cette fois sera satisfaite par l'antémémoire (cas du 470V/6 d'Amdhal),
- ou bien l'information venant de la mémoire centrale arrive directement dans l'unité d'instruction UI. L'antémémoire est alors remplie en parallèle (cas des DPS88) ou ultérieurement (cas des 470V/7, 470V/8, IBM 3033). Les américains appellent cette technique "fetch bypass", "load thru".

Notons aussi que selon les processeurs, les lectures ne sont envoyées qu'à l'antémémoire, ou bien simultanément à la mémoire et à l'antémémoire (et aux autres antémémoires dans les cas des multiprocesseurs, cas des DPS7): le premier qui répond est celui qui peut fournir l'information.

ORDINATEUR	NBR AM	TAILLE AM en Ko	TAILLE DU BLOC en o	NOMBRE DE COLONNES	NOMBRE DE NIVEAUX
VAX-11/780	1	8	8		2
VAX-11/750	1	4			
PRIME 850 (1)	1	16			
MV/8000	2		16	1024/64	1/1
1100/84	1	96 ou 128	16	256	4
3033N	1	16	64	64	16
168-3	1	32	32	128	8
3033U	1	64	64	164	16
3083	1		128		
Amdhal 5860	2	32 ou 64	32	128	8
3031	1	32	32	128	8
168-1	1			128	4
158-3	1	16		128	2
DPS88	2	32	64	256	4
DPS7	1	16	16	256	4
68020 (2)	1	0,256	4	64	1

(1) par processeur (2) contenue dans la puce elle-même

Figure 18m: exemples d'antémémoires.

18.8.2 - Algorithmes de placement et remplacement.

Ces algorithmes doivent choisir le bloc de l'antémémoire dans lequel sera chargé le bloc lu en MC. Ils relèvent généralement de deux grands types:
- le type FIFO (First In First Out, premier entré premier sorti) consiste à stocker le numéro du niveau attribué au dernier bloc chargé. Lorsqu'un nouveau bloc arrive, on lui attribue le numéro suivant. Il suffit ainsi d'un compteur modulo le nombre de niveaux de l'antémémoire pour gérer ce type d'algorithme, utilisé par les DPS8;
- le type LRU (Least Recently Used, le moins récemment utilisé) est

déjà plus complexe à implémenter car il tient compte de l'utilisation des blocs depuis leur chargement en antémémoire et cherche à en éliminer, lorsque cela devient nécessaire, ceux qui sont le moins utilisés.

Pour une antémémoire à deux niveaux, un seul bit suffira; pour n niveaux il faudra n(n-1)/2 bits par colonne. Cette croissance en fonction du carré de n devient trop onéreuse quand le nombre de niveaux dépasse 8. C'est pourquoi sur les 370/168-3 et les 3033 IBM a implémenté des LRU approchés ou **pseudo LRU.**

Sur les 168-3 les huit blocs d'une colonne sont groupés en 4 paires. On choisit d'abord la paire puis le bloc dans la paire et on a ainsi besoin de 10 bits au lieu de 28.

Sur le 3033 les blocs sont regroupés en quatre groupes de deux paires. Six bits sont nécessaires pour la recherche du groupe de deux paires de quatre blocs, quatre bits pour choisir une parmi deux paires dans un groupe, huit bits pour rechercher le bloc dans la paire, soit au total 18 bits au lieu des 120 requis par le LRU complet.

18.8.3 – Algorithmes de vidage.

Lorsque l'unité de commande émet une écriture il y a plusieurs façons de tenir compte de l'antémémoire et de la mémoire centrale:

- l'écriture se fait dans l'antémémoire puis dans la mémoire centrale (copy-back). Cette écriture peut d'ailleurs provoquer un remplissage de l'antémémoire si la donnée écrite n'est pas déjà chargée dans l'antémémoire (fetch on write);
- l'écriture en mémoire centrale est systématique (write thru, store thru) dès qu'une information est modifiée. On peut dans ce cas:
  * remplir l'antémémoire si elle ne contenait pas cette information,
  * mettre à jour l'antémémoire ou la vider si elle contient l'information. L'écriture en mémoire centrale avec vidage est appelée "write and invalidate" dans la littérature américaine.

  Les DPS8, VAX-11/780, 168-3, 3033, PDP-11/70 utilisent cette technique.
- l'écriture en antémémoire seulement (store in, write back, non write thru). La mémoire centrale est ainsi moins sollicitée, bien qu'à chaque vidage c'est tout un bloc et non pas quelques octets que l'on écrit. Les Amdahl 470, les 3081, 3083 d'IBM, les DPS88, les DPS7, SYNAPSE utilisent cette technique.

18.8.4 – Problèmes posés par la duplication.

La duplication de l'information dans la mémoire centrale et la ou les antémémoires pose un certain nombre de problèmes, dans les multiprocesseurs en particulier. Les solutions les plus répandues sont:

- la diffusion de toutes les écritures à toutes les antémémoires et à la mémoire centrale afin que toutes les copies soient toujours à jour. Ceci conduit à un trafic important;
- la diffusion des adresses des écritures à toutes les autres antémémoires afin qu'elles purgent les blocs correspondants (cas des

370/168 et 3033);
- faire en sorte que les données modifiables ne se trouvent que dans une seule mémoire à la fois (cas de certaines informations comme les mots sujets aux instructions de "test and set" qui vont toujours être manipulés dans la mémoire centrale dans les DPS8 sous GCOS3). Dans les DPS7 une donnée modifiée ne l'est que dans une seule antémémoire;
- interdire l'antémémoire aux données partageables modifiables (MULTICS sur DPS8M);
- disposer de répertoires centralisés ou décentralisés (cas du 3081).

18.8.5 – Nombre d'antémémoires.

Le nombre d'antémémoires est dans la plupart des cas de un par processeur, mais on trouve des machines à une antémémoire pour plusieurs processeurs (une pour deux sur le 1100/84 de Sperry). Dans d'autres cas on trouve plusieurs antémémoires par processeur, ou des antémémoires configurables en plusieurs parties.

Ainsi la MV/8000 de Data General dispose d'une première antémémoire (figure 18n) de 1024 blocs de 16 octets qui contient instructions et données.

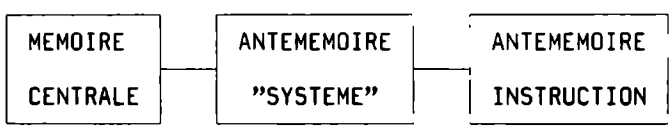


Figure 18n: les antémémoires de la MV/8000.

L'antémémoire instruction (instruction cache et system cache pour l'antémémoire du premier niveau) est très proche de ce que nous avons présenté sous le nom de tampon d'instructions, seule la gestion en est différente. D'autres processeurs disposent de deux antémémoires:
- l'une pour les **instructions**,
- l'autre pour les **données**.

Les caractéristiques des deux objets sont en effet différentes. Les instructions sont de plus en plus des objets non modifiables dans les processeurs modernes. Ceci facilite de plus l'anticipation de l'exécution des instructions: celle qui est dans sa phase d'appel sollicite l'antémémoire instruction, celle qui est en phase d'exécution a besoin de l'antémémoire données pour ses opérandes. Les deux antémémoires travaillent ainsi en parallèle grâce au pipelining des instructions.

On trouve ce découpage sur le Concept 32/67 de SEL, le 5860 d'Amdahl, l'ITEL AS/6, l'HITACHI H200, le Spectrum de HP, le DPS88 de Bull. Sur le microprocesseur Z800 de Zilog il est possible de configurer l'antémémoire pour qu'elle ne contienne que les instructions, que les données, ou bien les deux. L'antémémoire du 68020 ne concerne que les instructions.

Dans d'autres processeurs encore, c'est le mode de fonctionnement du processeur (maître/esclave) qui conduit à disposer de plusieurs antémémoires.

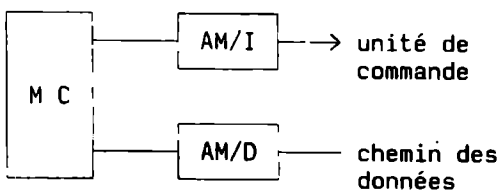


Figure 18o: deux antémémoires spécialisées, l'une pour les instructions et l'autre pour les données.

18.9 – Tampons mémoire.

L'antémémoire améliore le temps d'accès des mémoires centrales, elle ne joue un rôle qu'en lecture. Lorsque l'algorithme de vidage est du type "write thru" (écrire dans la mémoire centrale de façon systématique), type encore très répandu, le processeur est freiné lorsque le taux d'écriture en mémoire centrale croît.

Pour améliorer l'efficacité du processeur, on crée alors des tampons pour les opérandes à ranger en mémoire centrale. Ces tampons peuvent se trouver dans le processeur ou dans la mémoire elle-même.

Dans le cas du VAX-11/780, c'est l'unité de commande de la mémoire qui comporte un tampon pouvant stocker quatre requêtes.

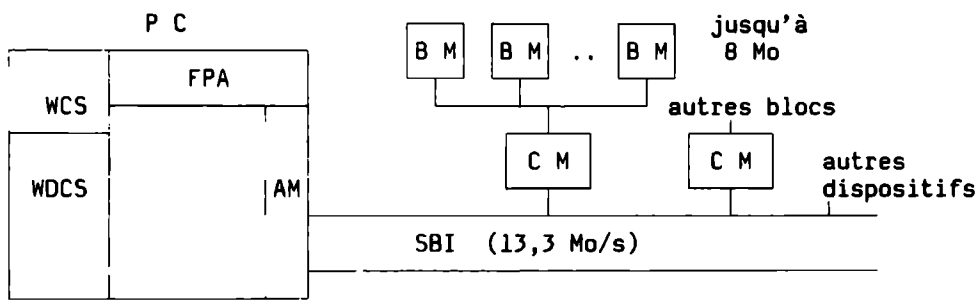


Figure 18p: configuration du matériel du VAX-11/780.

CM = unité de commande MC
BM = bloc mémoire de 256 Ko
FPA = option flottant câblé
WDCS, WCS = mémoire de commande
AM : antémémoire

18.10 – Mémoires étendues.

Lorsque l'on évoque les problèmes de taille mémoire, il y a deux notions à ne pas confondre:

- la taille de la mémoire centrale installée (on dit aussi **configurée**) dans un sous-système central donné,
- la taille de l'espace qu'une instruction peut adresser; on parlera **d'espace adressable** (address space).

L'espace adressable relève de l'architecture externe, la taille configurable maximum de l'architecture interne. Les séries 360/370 d'IBM ont un espace adressable de 16 Mo qui passe à 2 Go avec l'architecture étendue (XA). La taille configurable maximum d'un VAX-11/780 est de 8 Mo.

Les rapports entre mémoire configurable/configurée et espace adressable peuvent prendre différentes valeurs:

- l'espace adressable est **égal** à la taille maximum configurable: c'est souvent le cas des micro-ordinateurs 8 bits avec leurs 64 Ko, ce fut aussi le cas des minis 16 bits (PDP-11, ECLIPSE entre autres) avec leurs 64 Ko avant les extensions de mémoire. De tels systèmes sont souvent destinés à être mis en oeuvre par un seul utilisateur, caractéristique des micro-ordinateurs 8 bits;
- l'espace adressable est **inférieur** à la taille configurable: ce peut être une évolution du cas précédent, évidente dans le cas des minis 16 bits comme les PDP-11 qui peuvent avoir des mémoires centrales de plusieurs millions d'octets; plusieurs utilisateurs peuvent alors travailler simultanément sur la machine et disposer séparément de toute la mémoire que leur permet d'adresser l'architecture externe. Ils peuvent même, au moyen d'une technique de **bancs** (bank en américain), utiliser plus de mémoire que n'en offre l'espace adressable: il faut pour cela modifier de temps en temps les adresses des portions de mémoire sur lesquelles on travaille, c'est-à-dire **commuter** les bancs (bank switching), car bien entendu une instruction donnée ne peut adresser plus de zones mémoire que prévu par l'architecture externe. Au niveau du matériel, il faut donc introduire des dispositifs supplémentaires pour réaliser ce changement de la portion de mémoire adressée.

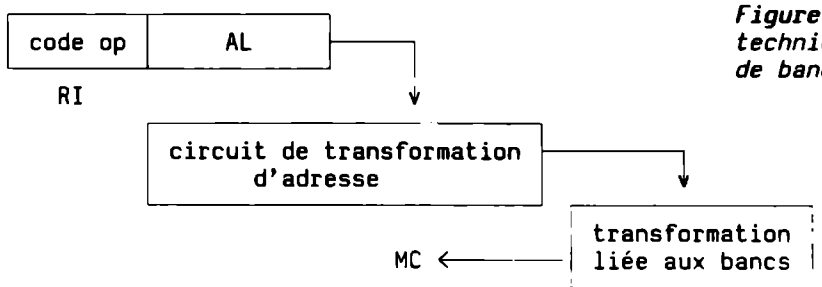


Figure 18q: technique de bancs.

Le banc "actif" peut être défini par un simple bit dans le cas de deux bancs. Il correspond à l'équivalent d'un registre de translation RT dans le cas où il y a plus de deux bancs qui peuvent débuter n'importe où en mémoire. Une généralisation de cette technique de bancs et son inclusion dans l'architecture externe est illustrée par le 8086:

* une première extension, par rapport aux 64 Ko d'un 8080, consiste à séparer code, donnée et pile; le code étant de plus en plus invariant, c'est-à-dire non modifié en cours d'exécution du programme, on suppose que les adresses liées aux instructions et aux données sont physiquement distinctes. D'où deux bancs (on dira **segments** dans la terminologie 8086 et dans celle de minis 16 bits qui ont connu la même évolution comme les MITRA de CII) de 64 Ko chacun. Ensuite la pile utilisée pour transmettre les arguments entre sous-programmes appelant et appelé et stocker les variables

locales à un sous-programme correspondra à un troisième segment appelé SS (stack segment) dans le 8086. Et sans doute parce que trois n'est pas une puissance de deux, le 8086 ajoute un quatrième segment appelé "extra segment" ES, segment supplémentaire, pour y ranger des données (une justification plus sérieuse est liée au fait qu'une chaîne destination est définie par ES:DI et qu'il faut pouvoir manipuler ds chaînes allant jusqu'à 64 Ko). On arrive ainsi à un espace adressable de 256 Ko, quatre fois 64 Ko;

* en modifiant le contenu des quatre registres de base de ces quatre segments, on peut adresser une mémoire configurable de 1 Mo. Mais pas simultanément, 256 Ko au plus à la fois. Ce chargement des registres de base des segments correspond à la commutation de bancs présentée plus haut.

Les registres de base des segments ont 16 bits, la taille du mot dans les 8086/8088, mais en fait ils définissent une adresse sur 20 bits, d'où les 1 Mo configurables, car quatre zéros leur sont systématiquement ajoutés (concaténés) à droite (figure 18r). Un segment commence ainsi toujours à une adresse multiple de 16 en mémoire centrale. En mettant la même valeur dans les registres de base de CS et DS on se rapprochera du fonctionnement d'un 8080 et il faut souligner que certains logiciels, comme des interpréteurs Basic sous MS-DOS, ne travaillent que dans 64 Ko.

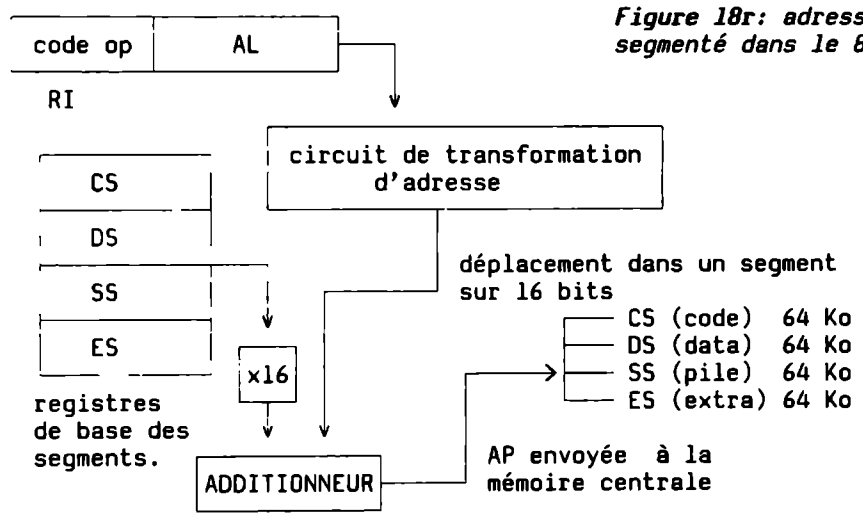


Figure 18r: adressage segmenté dans le 8086.

Il faut se méfier de certaines annonces trompeuses qui laissent à penser que toute la mémoire configurable appartient à l'espace adressable. Ces annonces sont fréquentes dans le domaine des multimicros 8 bits et une société américaine s'est faite condamner, il y a quelques années, par un client qui avait pris l'annonce au mot: on y annonçait qu'un multimicro de huit micro-ordinateurs disposait d'une mémoire de 512 Ko, mais il fallait comprendre que chacun des

huit micro-ordinateurs comportait 512/8 soit 64 Ko de mémoire centrale.

On peut mentionner dans ce paragraphe les extensions de mémoire que l'on trouve sur les DPS8 sous GCOS3. L'espace adressable y est de 256 Kmots (18 bits de déplacement dans les instructions) et l'espace réservé à un programme est de 256 Kmots moins celui occupé par le système d'exploitation GCOS3. A l'époque de la conception de cet ordinateur (GE600) 256 Kmots représentaient une taille énorme et plusieurs programmes pouvaient résider simultanément en mémoire centrale, le programme actif étant délimité par les registres de base (BAR, Base Address Register) et limite. Quand il fallut augmenter la taille des mémoires au-delà de 256 Kmots, on étendit tout simplement le registre de base BAR par un registre BER (Base Extension Register) qui permettait de disposer de plusieurs bancs (appelés quadrants dans la terminologie DPS8) de 256 Kmots, un programme devant obligatoirement se trouver à l'intérieur d'un banc et ne pouvant dépasser 192 Kmots (256 moins 64 mots, la taille maximale réservée à GCOS3) en dehors du premier quadrant.

- l'espace adressable est supérieur à la taille configurable: nous abordons là les machines à mémoire virtuelle que nous étudierons dans le Tome II.

Le terme de "mémoire étendue" s'emploie aussi pour des mémoires qui sont en fait accédées par des instructions d'entrée/sortie, comme des périphériques. C'est le cas de l'ECS (extended core) des CYBER de CDC, des mémoires d'arrière-plan des IBM 30XX.

Mentionnons encore la 1108 de Sperry qui avait une mémoire centrale composée de deux parties, l'une rapide en mémoire bipolaire, l'autre moins rapide en MOS. Cette structure n'a plus été reprise par la suite, les antémémoires apportant une solution plus astucieuse et moins problématique à gérer.

chapitre 19

fiabilité, disponibilité machines tolérant les pannes

19.1 – Fiabilité et disponibilité.

Nous n'avons guère évoqué jusqu'ici les problèmes de bon ou mauvais fonctionnement en supposant implicitement que l'ordinateur marchait correctement. En fait, construit avec des composants qui ont toujours des défauts, même infimes, les ordinateurs n'ont pas a priori un fonctionnement sans incident. Il en est de même de la plupart des constructions humaines, que ce soit les véhicules, les bâtiments, les avions, etc. Ces mauvais fonctionnements peuvent être surmontés (corrigés), le plus important étant d'ailleurs de les détecter afin d'arrêter si nécessaire l'ordinateur ou son sous-ensemble défaillant. Cette détection n'est pas toujours réalisée, tant s'en faut !

Pour quantifier l'aptitude d'un système à bien fonctionner on utilise sa **fiabilité**, définie selon [SIE84] comme une fonction F du temps t: F(t) est la probabilité pour que l'ordinateur fonctionne encore à l'instant t, après avoir fonctionné depuis l'instant zéro compris.

La fiabilité (reliability) d'un ordinateur, ou de l'un de ses sous-ensembles, peut être mesurée de deux façons:

- **théorique**, par le calcul, en utilisant la fiabilité de chacun de ses composants élémentaires. [ALL84] indique par exemple que le sous-système central du HP1000/A600 a été calculé pour un MTBF de 10 400 heures;
- **réelle**, en relevant les pannes qui se produisent sur les ordinateurs installés et en tirant des statistiques. [ALL84] précise que les pannes observées sur les HP1000/A600 donnent un MTBF deux fois plus grand que celui calculé.

La disponibilité est, quant à elle, une fonction D(t) du temps qui représente la probabilité pour que l'ordinateur fonctionne à l'instant t. On peut accroître la disponibilité d'un ordinateur:

- en utilisant des composants à haute fiabilité afin de réduire la probabilité d'un mauvais fonctionnement. C'est l'approche qui consiste à éviter les défauts (fault avoidance dans [SIE84]). Mais même les composants très fiables peuvent ne pas suffire, il faut alors prendre une autre approche,

- en utilisant la **redondance** afin de surmonter les erreurs inévitables, en les détectant et, si possible, en les corrigeant. C'est la technique dite "fault tolerant" qui conduit à "tolérer" les pannes inévitables.

Lorsque l'on parle de fiabilité et de disponibilité, on voit apparaître les acronymes MTBE, MTBF, MTTR:
- MTBE représente le temps moyen entre deux erreurs consécutives (mean time between errors);
- MTBF représente le temps moyen entre deux pannes consécutives (mean time between failure, certains emploient MTTF, mean time to failure), la différence ici entre erreur et panne étant que l'erreur peut être corrigée ("surmontée") alors que la panne ne peut pas l'être. Ce qui ne veut pas dire que l'ordinateur cesse de fonctionner globalement. Trilogy visait des processeurs avec un MTBF de trois à quatre ans;
- MTTD (mean time to detect) le temps pour que l'erreur soit détectée;
- MTTR représente le temps moyen pour réparer (mean time to repair).

Tous ces temps font généralement l'objet d'un engagement contractuel de la part du fournisseur. La disponibilité se définit par:

$$\text{disponibilité} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

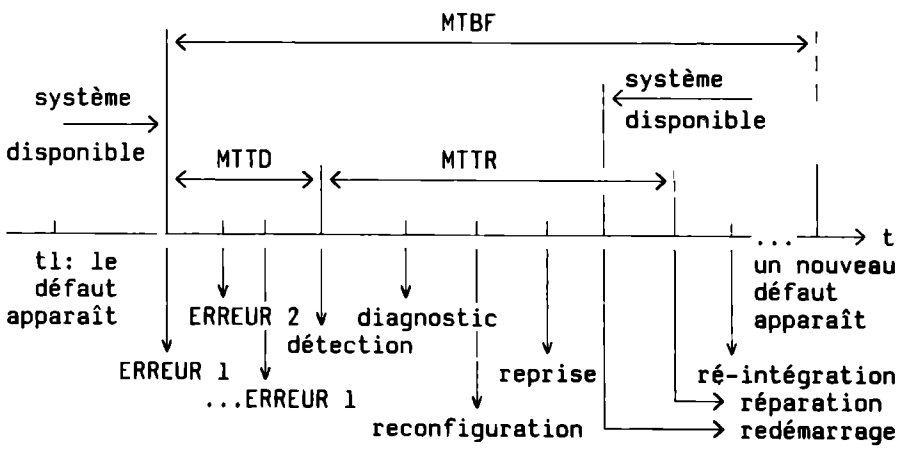


Figure 19a: les différentes étapes d'un mauvais fonctionnement selon [SIE84].

On distingue plusieurs types de panne, de défauts de fonctionnement:
- les pannes **permanentes** (solid faults), faciles à localiser puisque nettes. La localisation des défauts est un élément important au niveau de la **maintenance** des ordinateurs;

- les pannes **intermittentes** (intermittent faults), que l'on corrige en relançant plusieurs fois l'opération (retry, redondance temporelle);
- les pannes **fugitives** (transient faults) dont les symptômes apparaissent de façon très brève;
- les pannes **temporaires** (temporary faults) qui disparaissent lorsque l'ordinateur atteint un équilibre, de température par exemple.

En dehors des dispositifs architecturaux que nous allons présenter, la prévention des pannes est améliorée par la **maintenance préventive**. Si, il y a une dizaine d'années, cette maintenance, réalisée par le fournisseur, immobilisait l'ordinateur pendant une demi-journée chaque semaine, elle est devenue beaucoup moins fréquente. L'évolution générale du matériel et du logiciel a aussi permis de réaliser de nombreuses opérations de test ou de diagnostic (localisation des défauts) sans arrêter le fonctionnement normal de l'ordinateur: les tests peuvent se dérouler en même temps que les programmes normaux, il est possible d'isoler les composants que l'on veut tester (reconfiguration). De nombreux compteurs d'incidents sont gérés par le matériel et le système d'exploitation, et lorsqu'ils atteignent des seuils (threshold) prédéfinis, les tests sont lancés automatiquement et indiquent s'il faut ou non intervenir physiquement sur l'équipement fautif. Nous verrons des exemples dans le cas des mémoires centrales. De nombreux tests sont d'ailleurs écrits sous forme de micro-programmes (cf. WDCS, writable diagnostic control store du VAX-11/780 sur la figure 18p).

19.2 - Redondance: la parité.

Un des éléments architecturaux les plus utilisés pour améliorer la disponibilité, concerne la redondance. La redondance la plus simple est la **parité**, souvent utilisée dans les mémoires. Elle consiste à ajouter un bit supplémentaire à l'information dont on veut augmenter la probabilité d'intégrité. Il peut y avoir parité paire ou parité impaire. Supposons que nous utilisions une parité impaire: le bit supplémentaire sera alors choisi de telle façon que le nombre total de bits égaux à un soit impair (d'où le nom de parité impaire) dans l'information globale, bit de parité compris. Prenons un exemple avec des octets. Chaque octet occupera en fait neuf bits en mémoire centrale. Si on veut écrire 0001 0100, on écrira 0001 0100 1. Lorsque l'on relit cet octet, on compte le nombre de bits égaux à un. Si ce nombre est pair, il y a alors détection d'une **erreur de parité**. Selon le degré d'autonomie et de complexité de la mémoire, il y aura alors ou non une seconde, une troisième, une n-ième lecture. Si l'erreur ne se reproduit pas, le traitement continuera, mais un compteur sera incrémenté et lorsqu'il dépassera un seuil donné, la zone mémoire correspondante ne sera plus utilisée (cas de systèmes d'exploitation assez sophistiqués) jusqu'à ce que la mémoire soit remplacée, testée puis remise en service.

La parité ne permet que la détection d'un bit en erreur: elle ne permet pas de savoir quel est le bit erroné, donc de le corriger, et de plus elle ne voit pas d'erreur lorsque deux erreurs simultanées apparaissent. Pour illustrer ceci reprenons l'exemple précédent avec 0001 0100 1 écrit en mémoire et supposons que lors de la lecture on

obtienne 0101 0000 1, c'est-à-dire que le second bit (en partant de la gauche) est passé de zéro à un, et le sixième bit est passé de un à zéro; comme le nombre total de bits égaux à un est toujours impair, aucune erreur n'est détectée... Il en serait bien évidemment de même si c'était le bit de parité qui avait perdu sa valeur correcte: 0001 0000 0 ne provoquerait pas d'erreur non plus.

Souvent utilisée dans les mémoires centrales, quelquefois sur les bus (pratiquement jamais sur les bus données et adresses des microprocesseurs), très rarement dans les registres (cas du DCP40 de Sperry), la parité est aussi utilisée dans certaines mémoires de commande, et nous la retrouverons dans des sous-systèmes périphériques (bandes, communications asynchrones).

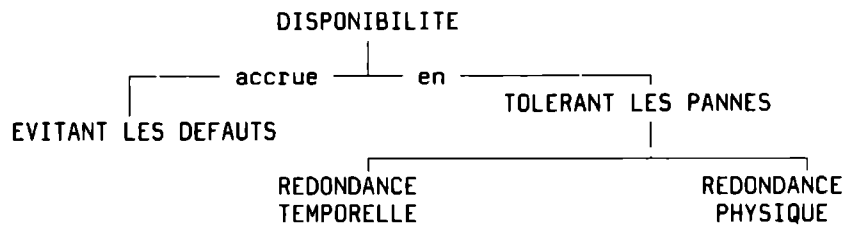


Figure 19b: classification des stratégies permettant d'accroître la disponibilité.

19.3 – Redondance: détection et correction d'erreur.

Une extension du simple bit de parité permet de surmonter les deux limites que nous avons citées:
- corriger les erreurs simples, c'est-à-dire les erreurs portant sur un seul bit,
- détecter les erreurs multiples, c'est-à-dire les cas où plus d'un bit est simultanément en erreur.

Les erreurs simultanées sur deux bits étant rares, on se limite généralement à leur détection et à la correction des erreurs sur un bit; on parlera alors d'ECC (error correcting code) ou d'EDAC (error detection and correction).

Les codes correcteurs d'erreur ont fait l'objet d'études volumineuses. Dans les mémoires centrales, le plus utilisé est le code de Hamming.

19.4 – Redondance: les différents types.

Les exemples ci-dessus ont permis d'illustrer deux types de redondance:
- **temporelle**, dans laquelle les actions sont répétées (retry) jusqu'à ce qu'une valeur limite soit atteinte ou que le défaut n'apparaisse plus. Ainsi le processeur de l'IBM 3083 fait jusqu'à huit tentatives avant de renoncer à exécuter l'instruction concernée. En cas d'échec il revient au dernier point de contrôle (checkpoint) fait au niveau matériel et reprend l'exécution dans l'espoir que le problème ne réapparaîtra pas;

- **physique**, en disposant de matériel supplémentaire, cas illustré par la parité et les codes correcteurs d'erreur.

Dès 1944, un ordinateur à relais de Bell exécutait deux fois chaque opération et comparait les résultats (redondance temporelle). L'UNIVAC I de 1951 utilisait les parités et deux UAL qui fonctionnaient en parallèle pour comparer leurs résultats (redondance physique, citée par [SIE84]).

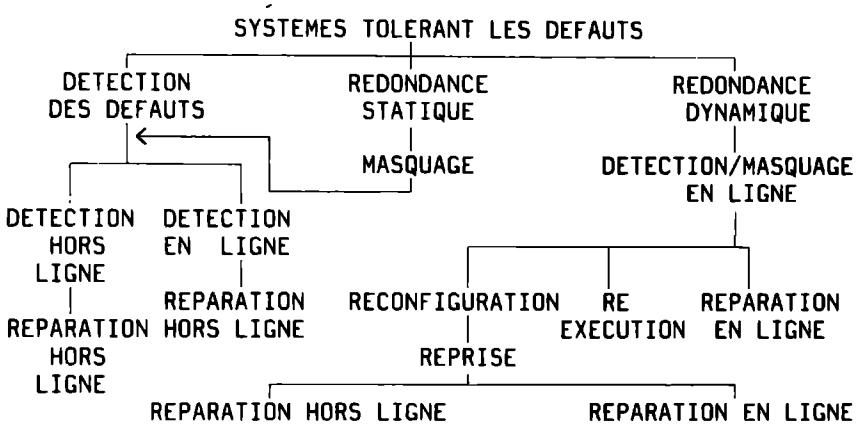


Figure 19c: classification des techniques utilisées dans les systèmes tolérant les défauts selon [SIE84].

19.5 – Systèmes doublés.

Une des redondances les plus classiques consiste à doubler le matériel, au niveau d'un opérateur (UAL par exemple) où les parités sont difficiles à mettre en oeuvre, et souvent plus coûteuses, au niveau d'un processeur, d'une mémoire, d'un périphérique... et même au niveau de l'ordinateur.

Dans le cas de l'ordinateur doublé complètement, on distinguera trois grands cas:

- la configuration **en secours** (back up) dans laquelle l'un des deux ordinateurs assure tout le travail et l'autre attend une panne éventuelle du premier. Pour signaler que tout va bien, le premier ordinateur envoie régulièrement des messages au second. Dès qu'il n'a plus d'informations suffisamment fraîches, le second reprend, à un niveau plus fin, le travail du premier (opération de basculement). Cette configuration est coûteuse puisqu'il faut doubler le matériel, ajouter des dispositifs de basculement (que l'on espère fiable!), alors que seule la moitié des ressources travaille utilement. On la rencontre fréquemment, lorsque l'on utilise des minis, dans des systèmes temps réel (contrôle de processus, machines de réseau, autocommutateurs comme l'OPUS 4000 ou le 3B-20D d'AT&T);
- la configuration **en partage de charge** (load sharing) dans laquelle les deux systèmes travaillent normalement. Si l'un des systèmes

tombe en panne, l'autre reprend sa charge, en offrant un service généralement **dégradé** puisqu'il supporte le double de la charge précédente. Cette solution est moins coûteuse que la précédente puisqu'en temps normal toutes les ressources travaillent de façon utile. En cas de défaut par contre le service offert devient moins bon. C'est une solution que l'on trouve de façon naturelle dans le cas des multiprocesseurs;
- la **redondance active**: configuration dans laquelle les deux systèmes font exactement les mêmes opérations, en comparant ou non les résultats. Lorsque l'un des deux systèmes tombe en panne, le second continue à assurer le service, sans dégradation, et sans retard apparent.

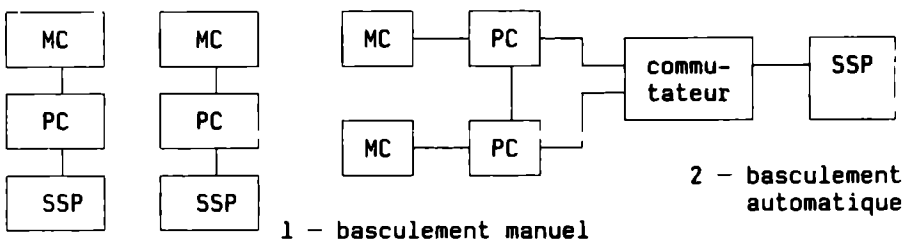


Figure 19d: configuration doublée du type "en secours".

19.6 – Les machines tolérant les pannes.

La première machine ainsi appelée fut celle de TANDEM: alors que les systèmes offrant le même type de service étaient généralement des **systèmes spéciaux**, c'est-à-dire des systèmes construits, adaptés pour les besoins d'un projet particulier, à partir de produits standard, TANDEM a mis sur le marché, en 1976, des systèmes conçus pour tolérer les pannes et proposés en standard dans son catalogue. Le modèle de bas de gamme avait tous ses composants doublés:
- toujours au moins deux processeurs centraux (en fait au moins deux systèmes), le maximum étant de 16. Chaque système a sa propre copie du système d'exploitation appelé Guardian;
- toute unité de commande de périphérique est reliée à deux processeurs centraux;
- tout périphérique est relié à deux "contrôleurs";
- toute écriture sur disque est double: on écrit sur deux disques différents en passant par deux chemins différents, d'où le nom de "disques miroirs";
- les alimentations sont doublées;
- le bus qui relie les différents systèmes (le DYNABUS à 10 Mo/s) est doublé.

La solution retenue dans les Tandem est intermédiaire entre les configurations en secours et en partage de charge, dans la mesure où un système n'est pas dédié à la surveillance de l'autre, mais chacun traite sa propre charge. Chaque programme (processus) est dupliqué, une copie dite "primaire" s'exécute dans un système et traite la

charge prévue pour ce programme. L'autre copie appelée "secours" se trouve obligatoirement dans un autre système, et se tient prête à reprendre la charge si le système du primaire tombe en panne. Afin d'être capable de reprendre cette charge, il reçoit régulièrement du primaire des messages de synchronisation définissant des points de contrôle (checkpoint). Cet échange de messages entre processus identiques réduit la puissance d'une paire de systèmes (cf. figure 19e) de 15 à 20 % selon [SIE84]. Chaque système envoie régulièrement (toutes les secondes) des messages convenus aux autres systèmes pour leur signaler qu'il fonctionne. S'il n'émet plus, les autres en concluent qu'il est tombé en panne.

Ces systèmes Tandem permettent des réparations en ligne, c'est-à-dire pendant le fonctionnement du reste du système global: on peut par exemple échanger des cartes sans arrêter le système. Cette machine est utilisée en général dans des environnements transactionnels (cf. Tome II).

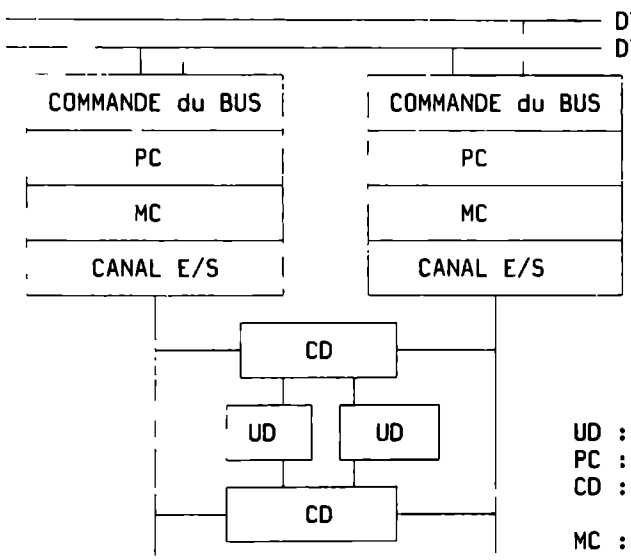


Figure 19e: structure générale d'un TANDEM dans sa configuration minimale.

UD : unité de disque
PC : processeur central
CD : contrôleur de disques
MC : mémoire centrale

Un autre type de machine tolérant les pannes est illustré par Stratus, commercialisé en France par Logabax, maintenant filiale d'Olivetti, sous le nom d'HYPER32. Stratus a démarré après Tandem, en 1980, et le premier système fut livré en 1982. Les fonctions principales y sont quadruplées:

- chaque carte (plaque) a sa carte équivalente en réserve (spare);
- chaque carte contient la fonction qu'elle assure sous forme doublée: chacun des deux circuits reçoit les mêmes entrées, et les sorties sont comparées afin de générer un signal d'erreur si elles ne sont pas identiques; ces comparaisons se font au rythme du cycle de base dans le cas des processeurs, soit huit millions de fois par seconde.

D'où le nom de "pair and spare" attribué à cette architecture. Si une erreur est détectée, la carte en réserve prend le relais, sans dégradation ni perte de temps apparentes. L'approche de Stratus est donc de type matériel alors que celle de Tandem requiert une complexité accrue du logiciel, jusqu'au niveau de l'application.

Cette approche "double et réserve" n'est utilisée que dans les processeurs et la mémoire centrale. Les unités de commande (contrôleurs) de disque ont des circuits de lecture et d'écriture doublés, et ils n'écriront pas sur les disques ou sur les bus système si les deux circuits ne produisent pas les mêmes sorties. Les disques peuvent être doublés si on le juge nécessaire et fonctionner alors en "disques miroirs". Les caractéristiques principales de l'HYPER32 sont:

- de 1 à 32 modules de traitement, chaque module comportant des 68000, un pour les programmes (processus) des utilisateurs et du système d'exploitation, un autre pour la gestion des interruptions (Tome III) et la pagination (Tome II);
- chaque module de traitement peut adresser 8 Mo de mémoire centrale; comme cette dernière est aussi doublée, elle atteint en fait 16 Mo. Les lectures et écritures y sont faites en double;
- le bus STRATALINK (HYPERLINK chez Logabax) à 1,4 Mo/s est doublé;
- les alimentations (power supply) sont aussi doublées.

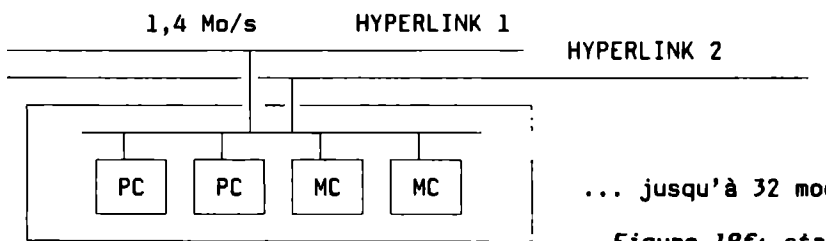


Figure 19f: structure générale d'un STRATUS.

Comme dans le cas de Tandem, les réparations peuvent être effectuées **en ligne.**

Les microprocesseurs iAPX 432 d'Intel comportent des dispositifs qui facilitent la construction de tels systèmes "pair and spare".

L'approche de Synapse, fondée en 1980, est du type multiprocesseurs (tightly coupled) et non pas multi-systèmes (loosely coupled) comme les deux précédentes (Tandem et Stratus). Elle est appelée "N+1" car pour assurer une redondance, on ajoute un processeur aux N processeurs requis pour traiter la charge prévue. Le matériel supplémentaire est donc beaucoup plus limité que dans les approches de Tandem et Stratus qui sont du type 2N (ou N+N). Les caractéristiques des systèmes Synapse sont:

- jusqu'à 28 processeurs, basés sur des 68000; les uns sont spécialisés dans le traitement des entrées/sorties, les autres déroulent les applications et vont eux-mêmes chercher du travail (dispatching qui sera vu dans le Tome II) dans la mémoire centrale;
- la mémoire centrale de 16 Mo est commune à tous les processeurs. Elle n'est pas doublée. Pour des raisons de performance ces derniers

ont des antémémbires de 16 Ko;
- processeurs et mémoire sont interconnectés par un bus de 64 Mo/s doublé, de 32 bits de large.

En cas de panne d'un processeur, sa charge est assurée par les N processeurs restants. En cas de panne de la mémoire centrale, tous les processeurs s'arrêtent, le système d'exploitation est rechargé, la base de données est reconstituée, et le système repart après une interruption de quelques minutes. Les utilisateurs ont au maximum à re-saisir le dernier écran, grâce à un système interne de points de contrôle ([SER84]).

SYSTEME	NBR BITS	TYPE	NBR MODULE	TOPO-LOGIE	DISQUE MIROIR	1-ERE LIVRAI.
AUGUST	16	8086	3		non	10.81
AURAGEN	32	MS,68010	32	B		
HP 1000	16	MS,s	2		P	
IBM Série/1	16	MS,s		A	P	
STRATUS	16/32	MS,68000	32	B	P	03.82
SYNAPSE	16/32	MP,68000	28	B	L	08.82
TANDEM I	16	MS,s	16	B	P	05.76
TANDEM II	16/32	MS,s	16	B	P	04.81
TANDEM TXP	32	MS,s	16*	B	P	11.83
TOLERANT SYSTEMS		MS,NS32032		B		05.84
VAX Cluster	32	MS,s	<15	E	oui	

* : par bus MS : multisystèmes MP : multiprocesseur
s : jeu d'instructions spécifique B : bus A : anneau
L : logique P : physique E : étoile

Tableau 19g: quelques systèmes tolérant les pannes.

Le tableau partiel 19g montre que le créneau des systèmes tolérant les pannes est maintenant très fourni, trop même car les estimations de marché ont été exagérées et les premières faillites sont apparues dès 1985.

Nous avons vu jusqu'ici des systèmes basés principalement sur un **doublement** des ressources, avec éventuellement une comparaison des résultats. Il existe aussi des systèmes à **vote majoritaire** dans lesquels plus de deux systèmes font les mêmes opérations, comparent leur résultat, et en cas de discordance le résultat de la majorité l'emporte (August Systems par exemple). Les systèmes qui pilotent les fusées spatiales sont ainsi fréquemment triplés, voire quadruplés.

19.7 - Autres dispositifs.

Sur les systèmes classiques, les problèmes de fiabilité et de disponibilité ont fait apparaître des **processeurs de service** comme le SURP sur les DPS7, le sous-système console du VAX-11/780, le SSU (system support unit) sur les DPS8, le "support processor" des IBM 43XX.

Ces dispositifs permettent de dérouler des tests et des diagnostics (T&D) en ligne, d'intervenir **à distance** (remote maintenance, remote support facility). Ils disposent parfois de chemins séparés vers les autres composants de l'ordinateur.

Sans entrer dans le détail, on trouvera encore les éléments suivants qui contribuent à une meilleur disponibilité:

- les **batteries** qui font parfois l'objet d'une option et peuvent se substituer, pendant un temps plus ou moins long, à l'alimentation secteur;
- les mécanismes de **protection mémoire** que nous étudierons dans le Tome II: ils interdisent l'accès à certaines cases mémoire, en écriture, lecture ou exécution, selon le mode de fonctionnement (maître/esclave);

chapitre 20

les autres architectures

Nous avons jusqu'ici présenté des architectures "conventionnelles", les plus nombreuses. Il n'est pas toujours évident de caractériser une architecture de "nouvelle", tout comme pour les "nouveaux philosophes" ou la "nouvelle cuisine française"; il faut tenir compte des effets de mode, de vocabulaire, de présentation. Sans prendre parti, nous nous contenterons de dresser une liste des tendances nouvelles, originales, marginales, dans l'architecture des sous-systèmes centraux.

20.1 – Architectures conventionnelles: von Neumann.

Il importe de définir plus précisément ce que l'on entend par architecture conventionnelle. Elle est liée au nom de **von Neumann**, ingénieur hongrois qui définit en 1946 les quatre unités principales nécessaires à un ordinateur:

- l'unité de commande,
- la mémoire,
- l'opérateur (UAL),
- l'unité d'entrée/sortie.

Cette architecture est aussi dite "de Princeton", université où exerçait von Neumann. Un **seul processeur** y travaille en série (de façon séquentielle) sur des données qu'il va chercher en **mémoire** centrale (qui joue un rôle passif) pour les faire traiter par l'UAL, puis ranger les résultats dans cette même mémoire. La mémoire sert aussi bien à stocker les données que les programmes qui les traitent. On l'appelera parfois mémoire "globale" pour traduire ce fait et on utilisera aussi l'expression de machine à "programme enregistré" (stored program computer).

Une autre façon de caractériser l'architecture de von Neumann concerne la présence du compteur ordinal CO dont l'évolution du contenu définit la séquence dynamique des instructions. Ce compteur unique implique l'existence d'une architecture **centralisée** autour d'un centre de commandement unique, ce qui constitue un frein au traitement parallèle.

Les machines avec programme enregistré, et non plus lu sur ruban, sont apparues dès 1948 avec la SSEC (Selective Sequence Electronic Calculator) d'IBM: elle permettait les répétitions et les alternatives.

20.2 – Architecture de Harvard.

Lorsque les programmes et les données disposent de mémoires distinctes on parle d'architecture de Harvard, ou de **Aiken**, professeur exerçant à Harvard. Les tout premiers ordinateurs électromécaniques et électroniques suivaient cette architecture: les programmes étaient stockés sur des rubans perforés et les données étaient matérialisées par des relais téléphoniques à dix pas.

Les microprocesseurs PPS-8 de Rockwell, 4004 d'Intel relèvent de cette architecture. Il en était de même du Mark I de Harvard, en 1944, un monstre de 17 mètres de long, 25 mètres de haut, avec 18 000 tubes radio.

20.3 – Architecture avec cadencement des données.

L'architecture conventionnelle offre un frein intrinsèque au parallélisme, bien que ses effets puissent en être atténués par les mécanismes de parallélisme et d'anticipation présentés dans les chapitres 17 et 18. L'architecture à cadencement des données (**data-flow**, data-driven architecture) a pour but d'exploiter, au niveau de l'architecture, le parallélisme intrinsèque inhérent aux programmes. On peut la classer comme un cas particulier de MIMD (flot multiple d'instructions et flot multiple de données).

Ce terme de "flots de données" s'oppose aux architectures conventionnelles qualifiées d'architectures à "flots d'instructions" (instruction flow, control-driven).

Dans une machine à cadencement des données, il n'y a plus de compteur ordinal ou d'équivalent. Il n'y a pas d'unité de commande, ni de mémoire "globale" comme dans l'architecture de von Neumann. Une instruction peut y être exécutée dès que ses données sont prêtes, c'est-à-dire dès que ses opérandes ont pris la valeur voulue. Le résultat qu'elle va créer va permettre de débloquer, partiellement ou totalement, une ou plusieurs (parallélisme intrinsèque) instructions, et ainsi de suite. Ce sont les **valeurs** des données, et non plus leurs **noms** ou leurs adresses qui interviennent.

Le seul aspect séquentiel d'un programme est celui qui y est mis de façon **explicite** par le programmeur, ou créé par les ressources physiques limitées de la machine. Un programme n'est généralement pas présenté sous forme de liste séquentielle d'instructions, mais sous forme de graphe orienté avec comme noeuds les étapes élémentaires du traitement; sur les arêtes circulent des valeurs appelées **jetons** (token en américain). Exécuter une instruction revient donc à:

- absorber ses jetons d'entrée,
- faire exécuter la fonction par une unité fonctionnelle,
- générer un ou plusieurs jetons (valeurs de sortie) placés sur les arêtes sortantes.

Le pipelining à l'intérieur de chaque noeud élémentaire, ainsi qu'au niveau de l'ensemble du programme permet d'accroître encore la simultanéité.

Un des gros problèmes posés par cette architecture est celui des langages de programmation.

Les applications visées sont le calcul scientifique, les traitements symboliques (l'intelligence artificielle). Il existe de nombreux projets de machines à CdD (cadencement des données) de par le monde:
- le système LAU en France,
- SIGMA-1, DDDP, EDDY, etc. au Japon (cf [TOS84]).

20.4 – Architectures et langages de haut niveau.

Lorsque l'on aborde le sujet de l'architecture externe et des langages de haut niveau, il faut distinguer deux grandes approches:
- la première consiste à souligner que l'architecture externe implémente des concepts que l'on retrouve dans tel ou tel langage. Ainsi les piles du B5000 facilitent la gestion de la mémoire centrale pour le passage des paramètres et le stockage des variables locales des procédures écrites en ALGOL. L'iAPX 432 est souvent présenté comme une machine inspirée par ADA, le NS 32000 le serait par Pascal, le Bellmac-32 par le C, etc.
  Le matériel facilite et améliore l'exécution des programmes écrits dans un tel langage, mais il n'est pas question de supprimer les compilateurs;
- la seconde approche concerne ce problème de compilation qui disparaît, la machine ayant comme langage machine un langage de haut niveau, c'est-à-dire que son matériel, ses circuits et/ou ses microprogrammes, interprètent directement ce langage.

Les **avantages** espérés sont:
- un gain de temps grâce à la suppression de la compilation, et
- grâce à l'exécution plus rapide,
- des programmes plus denses, occupant moins de place en mémoire centrale.

Comme **inconvénients** nous avons:
- la nécessité d'avoir de gros microprogrammes pour améliorer l'interprétation,
- le problème posé par la sauvegarde des résultats intermédiaires en cas d'interruption,
- la difficulté d'aboutir à une solution favorisant également plusieurs langages si on désire une machine suffisamment générale (universelle).

Western Digital a ainsi microprogrammé différemment le LSI-11 pour lui faire exécuter directement du P-code ("Pascal Microengine"), forme intermédiaire générée par de nombreux compilateurs Pascal: on se trouve ici à mi-distance entre les vraies machines langage et les machines inspirées par un langage particulier. Le Transputer d'INMOS est dans un cas analogue puisqu'il exécute directement la forme intermédiaire de OCCAM, langage dans la mouvance de Pascal et ADA. Novix a annoncé une machine FORTH basée sur le microprocesseur microprogrammable NCR32.

La vogue actuelle de l'intelligence artificielle a remis en selle un vieux langage comme LISP et dès 1982 des machines LISP ont été introduites sur le marché américain (Xerox, LMI, Symbolics). Toujours dans le domaine de l'intelligence artificielle, le langage Prolog inspire un certain nombre de projets de par le monde (Maïa en France, Alice en Angleterre, plusieurs projets au Japon).

20.5 – Architectures RISC.

Nous avons déjà présenté les grands traits des architectures basées sur un jeu d'instructions réduit (de 30 à 50 instructions). Elles font d'ailleurs partie des architectures conventionnelles, et apparaissent pour certains comme le retour vers un âge d'or perdu (cf. [CAR85]). Les anciennes machines de CDC comme le 6600 ne sont d'ailleurs pas éloignées des concepts RISC. 1985 voit le triomphe de ces architectures et, la mode facilitant les choses, de nouvelles techniques qui n'ont rien à voir avec le RISC en utilisent la bannière.

Ainsi Pyramid qui annonce en 1983 le 90X comme étant la première implémentation commerciale d'une architecture RISC est-elle accusée par certains puristes de faire appel au jeu d'instructions conventionnel plus complexe (CISC pour complex instruction set computer, opposé à RISC) pour répondre aux problèmes du flottant et des entrées/sorties ([CAR85]). On s'attend à une floraison de RISC: RISC total, RISC partiel, etc.

L'approche RISC est initialement une approche universitaire qui cherche à optimiser le processeur central en prenant en compte le fait qu'une petite partie du jeu d'instructions représente la majeure partie des instructions exécutées; les chargements (load), appels (call) et branchements représentent couramment 60 à 80% du temps d'exécution. Il n'est pas certain qu'elle soit bonne pour les applications de gestion où le Cobol et les entrées/sorties dominent. Les comparaisons entre RISC et non-RISC (CISC) ont d'ailleurs porté principalement sur UNIX et son langage C, sur Pascal.

En dehors du jeu d'instructions réduit, les grandes caractéristiques des machines RISC sont:

- un nombre important de **registres** (528 sur Pyramid 90X), ce qui fait tomber le nombre d'accès à la mémoire centrale, et facilite le travail des compilateurs. [BAS85] justifie cependant les 16 registres du Ridge 32 par les travaux de J. Hennessy qui montrent que 8 registres suffisent pour la plupart des procédures;
- en dehors des instructions d'accès à la mémoire **un cycle** suffit pour leur exécution. Ceci simplifie le décodage et facilite l'anticipation (le pipelining) dans le traitement des instructions;
- toutes ont la même longueur, le **même format**, le code opération et les champs adresse occupent toujours les mêmes positions. Le Ridge 32 dispose cependant de 3 formats d'instruction selon que le déplacement occupe **zéro, seize ou trente deux bits**;
- l'approche relève du **câblé** et non pas de la microprogrammation.

20.6 – Architecture multimicros et multibus.

Le fait de disposer directement de petits processeurs (les microprocesseurs) permet de créer facilement des micro-ordinateurs, sans avoir à concevoir un nouveau jeu d'instructions, une nouvelle architecture externe, tâche qui n'est pas à la portée de tous.

Les microprocesseurs ayant des puissances limitées à un moment donné, lorsque l'on désire obtenir des puissances importantes, il faut:

- utiliser des microprocesseurs en tranches,
- disposer plusieurs processeurs en parallèle,
- combiner ces deux techniques.

Certains de ces multimicros, les premiers commercialisés en particulier, étaient en fait constitués par plusieurs micro-ordinateurs indépendants les uns des autres, supervisés par un micro-ordinateur central qui gérait les fichiers centraux. Ce type de structure se retrouve encore sur les Wang OIS (machines à vocation bureautique, OIS signifiant Office Information System) où chaque poste de travail est un micro-ordinateur avec éventuellement des disquettes de sauvegarde, les disques durs étant gérés par un micro-ordinateur central.

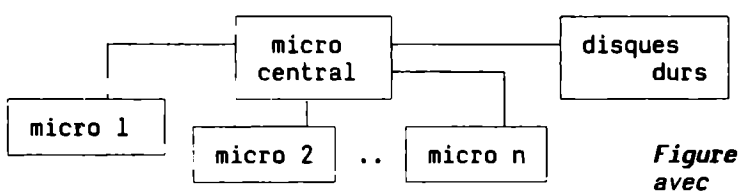


Figure 20a: multimicro avec micro central.

Dans ces systèmes, le seul élément de communication entre les différents postes de travail est constitué par les fichiers que gère le micro central. Les grappes de terminaux classiques, de nombreuses machines bureautiques sont ainsi conçues. Si un poste de travail n'est pas utilisé il constituera une ressource non utilisée; il n'y a aucune vue d'ensemble et le micro central répond aux demandes de chaque micro, sans pour cela les gérer en tant que ressources communes.

Lorsque l'on recherche des performances accrues en mettant plusieurs micros en parallèle il y a alors une certaine vue d'ensemble qui s'impose et elle conduit à une distribution horizontale et/ou verticale des micros.

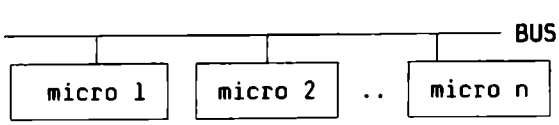


Figure 20b: distrition horizontale des micros le long d'un bus.

La distribution **horizontale** se fait généralement le long d'un bus, comme dans certains systèmes à tolérance de panne que nous avons

présentés dans le chapitre précédent. Chacun des micros peut être spécialisé (cas de commutateurs de paquets TP4000 de GTE-Telenet où un micro joue un rôle de pilote de l'ensemble et les autres gèrent des lignes de communication, tous les micros étant des 6502).

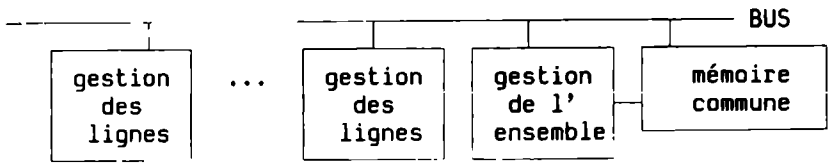


Figure 20c: distribution horizontale avec micros spécialisés (TP4000).

Les micros, ou certains sous-ensembles peuvent aussi être **banalisés** (cas des SYNAPSE dits "N+1").

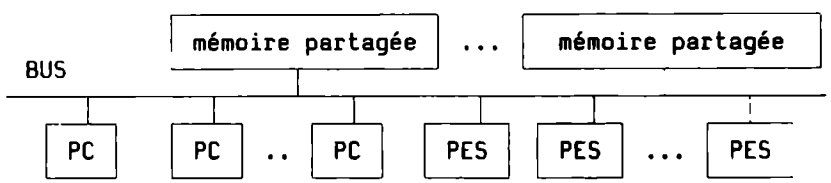


Figure 20d: distribution horizontale avec pools de micros banalisés (pool de processeurs centraux, pool de processeurs d'entrée/sortie) dans le cas de SYNAPSE N+1.

Cette spécialisation des micros est faite au niveau du matériel et/ou du logiciel. Les microprocesseurs ne sont pas forcément **homogènes** et peuvent très bien être **hétérogènes**: c'est un des objectifs du multi-micro SM90 conçu par le CNET et commercialisé par plusieurs sociétés françaises comme Bull (SPS6), CSEE (Coralis 90), Telmat, TRT (NPX90).

La distribution **verticale** apparaît lorsque les différents microprocesseurs se transforment en modules plus puissants et plus indépendants. On aura ainsi un premier étage raccordé au bus qui traite les fonctions de haut niveau (gestion de l'ensemble, commutation dans un commutateur de paquets), et un second étage qui réalise des fonctions plus élémentaires comme la gestion d'un protocole de communication. C'est ce que l'on trouve par exemple dans un DATEM de SESA ou un NPX90 de TRT.

Chaque module peut ainsi réaliser des fonctions plus complexes, ce qui diminue les communications avec les autres modules et décharge le bus. Le premier étage comporte généralement un microprocesseur 16 bits (68000 chez TRT) ou un 8 bits "de course" (Z80 à 8 Mhz chez SESA), le second étage contient un microprocesseur 8 bits, ou un

microprocesseur en tranches pour des protocoles gérant des communications rapides (AMD chez SESA), ou encore un circuit spécialisé dans la gestion d'un protocole particulier (X.25 chez TRT). Le nombre d'étages n'est bien évidemment pas limité à deux.

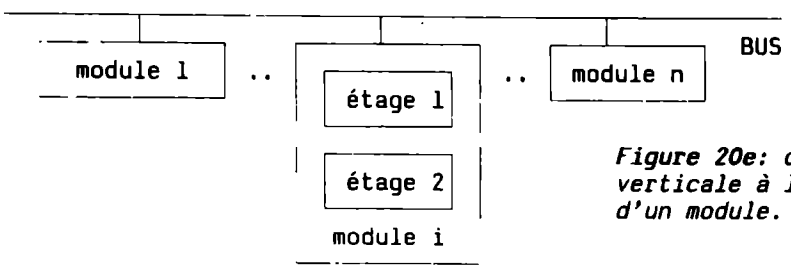


Figure 20e: distribution verticale à l'intérieur d'un module.

Nous avons fait allusion au problème posé par la charge du bus. Le bus constitue en effet un frein lorsque le nombre de microprocesseurs et les communications entre processeurs augmentent. Une solution consiste à réduire autant que possible la communication interprocesseurs ou entre processeurs et mémoire commune. Chaque processeur se voit alors doté d'une **mémoire locale.**

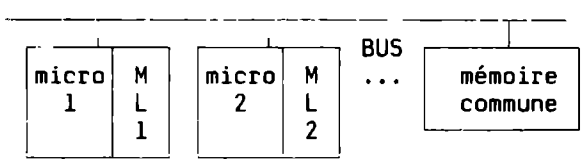


Figure 20f: mémoires locales déchargeant le bus.

MLi : mémoire locale i

Une des solutions les plus simples consiste à mettre dans ces mémoires locales les programmes que doivent dérouler les différents processeurs. Le bus sera déchargé d'autant et les processeurs travailleront plus vite car ils n'auront pas à attendre que le bus soit disponible pour charger les instructions. C'est la solution retenue dans les TP4000, les Coralis 80 de CSEE, etc. Dans les systèmes temps réel où la charge est connue à l'avance, les programmes eux-mêmes sont stockés dans des mémoires mortes locales. Cette structure est appelée "hybride" dans [KLE85].

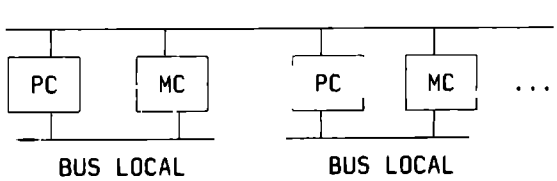


Figure 20g: hiérarchie de bus pour décharger le bus intermodules.

Une autre méthode pour décharger le bus interprocesseurs ou intermodules conduit à utiliser une **hiérarchie de bus,** le premier niveau est utilisé pour les relations intermodules, un second niveau pour les relations intramodules. Il peut même y avoir d'autres

niveaux, pour les entrées/sorties par exemple. C'est ce que l'on retrouve dans le X83 de Thomtit, c'est un des objectifs du Multibus II d'Intel que nous étudierons dans le Tome III.

Quand le nombre de modules raccordés sur un même bus devient trop important, il apparaît des problèmes de performance (la vitesse maximale du bus décroît), de coût (les petites configurations sont pénalisées), de sécurité (le bus constitue l'élément le plus faible de l'ensemble). Les concepteurs de multimicros évoluent alors vers des solutions comportant plusieurs bus intermodules. C'est le cas des X83 de Thomtit, des Engine dans Sophonet de Philips. L'Engine ne comporte pas de mémoire commune, il correspond à l'architecture appelée "à passage de messages" de [KLE85]. Le X83 combine multibus et hiérarchie de bus comme le montre la figure 20h.

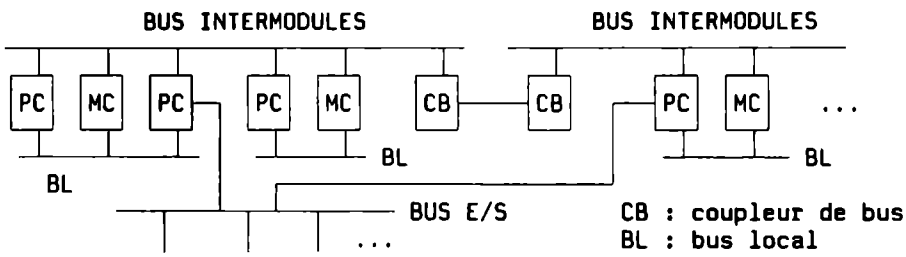


Figure 20h: multibus intermodules avec hiérarchie de bus (cas du X83).

Ces multimicros deviennent ainsi de véritables réseaux avec des chemins multiples entre les différents microprocesseurs et mémoires. On trouve aussi des solutions du type multimicros multibus dans certains ordinateurs cellulaires évoqués au chapitre 17.

20.7 – Autres architectures.

Nous avons déjà évoqué les architectures orientées objet.

Les architectures associatives se retrouvent dans des parties de l'ordinateur, mais pratiquement jamais au niveau global. Nous les avons illustrées avec l'antémémoire, nous les retrouverons dans la traduction d'adresse (Tome II), dans les disques à structure CHR (Tome IV).

annexe A

mnémoniques du 6502

	code	mnémonique	lg.	cyc.	indicateurs N	Z	C	I	D	V	
69	01101001	adc#	2	2	*	*	*			*	(A)+(C)+opérande -> A
65	01100101	adcz	2	3	*	*	*			*	
75	01110101	adczx	2	4	*	*	*			*	
6D	01101101	adca	3	4	*	*	*			*	
7D	01111101	adcax	3	4x	*	*	*			*	
79	01111001	adcay	3	4x	*	*	*			*	
61	01100001	adcx*	2	6	*	*	*			*	
71	01110001	adc*y	2	5x	*	*	*			*	
29	00101001	and#	2	2	*	*					(A) ET opérande -> A
25	00100101	andz	2	3	*	*					
35	00110101	andzx	2	4	*	*					
2D	00101101	anda	3	4	*	*					
3D	00111101	andax	3	4x	*	*					
39	00111001	anday	3	4x	*	*					
21	00100001	andx*	2	6	*	*					
31	00110001	and*y	2	5	*	*					
0A	00001010	asl	1	1	*	*	*				C <- A <- 0
06	00000110	aslz	2	5	*	*	*				
16	00010110	aslzx	2	6	*	*	*				
0E	00001110	asla	3	6	*	*	*				
1E	00011110	aslax	3	7	*	*	*				
90	10010000	bcc	2	2@							brancher si C = 0
B0	10110000	bcs	2	2@							brancher si C = 1
F0	11110000	beq	2	2@							brancher si Z = 1
24	00100100	bitz	2	3	*	*				*	Z=1 si (A) ET opérande=0
2C	00101100	bita	3	4	*	*				*	Z=0 sinon N<-bit de gauche de l'opérande V<-2e bit de gauche de l'opérande
30	00110000	bmi	2	2@							brancher si N = 1
D0	11010000	bne	2	2@							brancher si Z = 0
10	00010000	bpl	2	2@							brancher si N = 0
00	00000000	brk	1	7				1			((C0)+2) -> pile, (CC) -> pile
50	01010000	bvc	2	2@							brancher si V = 0
70	01110000	bvs	2	2@							brancher si V = 1
18	00011000	clc	1	2	0						0 -> C
D8	11011000	cld	1	2					0		0 -> D
58	01011000	cli	1	2				0			0 -> I
B8	10111000	clv	1	2						0	0 -> V
C9	11001001	cmp#	2	2	*	*	*				compare (A) avec l'opérande
C5	11000101	cmpz	2	3	*	*	*				
D5	11010101	cmpzx	2	4	*	*	*				
CD	11001101	cmpa	3	4	*	*	*				
DD	11011101	cmpax	3	4x	*	*	*				

code		mnémonique	lg.	cyc.	indicateurs N	Z	C	I	D	V	
D9	11011001	cmpay	3	4x	*	*	*				
C1	11000001	cmpx*	2	6	*	*	*				
D1	11010001	cmp*y	2	5x	*	*	*				
E0	11100000	cpx#	2	2	*	*	*				compare (X) avec l'opérande
E4	11100100	cpxz	2	3	*	*	*				
EC	11101100	cpxa	3	4	*	*	*				
C0	11000000	cpy#	2	2	*	*	*				compare (Y) avec l'opérande
C4	11000100	cpyz	2	3	*	*	*				
CC	11001100	cpya	3	4	*	*	*				
C6	11000110	decz	2	5	*	*					Opérande - 1 -> opérande
D6	11010110	deczx	2	6	*	*					
CE	11001110	deca	3	6	*	*					
DE	11011110	decax	3	7	*	*					
CA	11001010	dex	1	2	*	*					(X) - 1 -> X
88	10001000	dey	1	2	*	*					(Y) - 1 -> Y
49	01001001	eor#	2	2	*	*					(A) **OU EXLUSIF** opérande -> A
45	01000101	eorz	2	3	*	*					
55	01010101	eorzx	2	4	*	*					
4D	01001101	eora	3	4	*	*					
5D	01011101	eorax	3	4x	*	*					
59	01011001	eoray	3	4x	*	*					
41	01000001	eorx*	2	6	*	*					
51	01010001	eor*y	2	5x	*	*					
E6	11100110	incz	2	5	*	*					opérande + 1 -> opérande
F6	11110110	inczx	2	6	*	*					
EE	11101110	inca	3	6	*	*					
FE	11111110	incax	3	7	*	*					
E8	11101000	inx	1	2	*	*					(X) + 1 -> X
C8	11001000	iny	1	2	*	*					(Y) + 1 -> Y
4C	01001100	jmp	3	3							((CO)-1) -> COd, ((CO)+2) -> COg
6C	01101100	jmp*	3	5							
20	00100000	jsr	3	6							((CO)+2) -> pile,((CO)+1) -> COd, ((CO)+2) -> COg
A9	10101001	lda#	2	2	*	*					Opérande -> A
A5	10100101	ldaz	2	3	*	*					
B5	10110101	ldazx	2	4	*	*					
AD	10101101	ldaa	3	4	*	*					
BD	10111101	ldaax	3	4x	*	*					
B9	10111001	ldaay	3	4x	*	*					
A1	10100001	ldax*	2	6	*	*					
B1	10110001	lda*y	2	5x	*	*					
A2	10100010	ldx#	2	2	*	*					Opérande -> X
A6	10100110	ldxz	2	3	*	*					
B5	10110101	ldxzy	2	4	*	*					
AE	10101110	ldxa	3	4	*	*					
BE	10111110	ldxay	3	4x	*	*					
A0	10100000	ldy#	2	2	*	*					Opérande -> Y
A4	10100100	ldyz	2	3	*	*					
B4	10110100	ldyzx	2	4	*	*					
AC	10101100	ldya	3	4	*	*					
BC	10111100	ldyax	3	4x	*	*					
4A	01001010	lsr	1	2	*	*					0 -> A -> C
46	01000110	lsrz	2	5	*	*	*				
56	01010110	lsrzx	2	6	*	*	*				
4E	01001110	lsra	3	6	*	*	*				
5E	01011110	lsrax	3	7	*	*	*				
EA	11101010	nop	1	2							fonction nulle
09	00001001	ora#	2	2	*	*					(A) **OU** opérande -> A
05	00000101	oraz	2	3	*	*					
15	00010101	orazx	2	4	*	*					
0D	00001101	oraa	3	4	*	*					
1D	00011101	oraax	3	4x	*	*					
19	00011001	oraay	3	4x	*	*					
01	00000001	orax*	2	6	*	*					
11	00010001	ora*y	2	5	*	*					

	code	mnémonique	lg.	cyc.	indicateurs N	Z	C	I	D	V	
48	01001000	pha	1	3							(A) dans la pile
08	00001000	php	1	3							(CC) dans la pile
68	01101000	pla	1	4	*	*					A <- pile
28	00101000	plp	1	4	de la pile						CC <- pile
2A	00101010	rol	1	2	*	*	*				C <- A <- C
26	00100110	rolz	2	5	*	*	*				
36	00110110	rolzx	2	6	*	*	*				
2E	00101110	rola	3	6	*	*	*				
3E	00111110	rolax	3	7	*	*	*				
6A	01101010	ror	1	2	*	*	*				C -> A -> C
66	01100110	rorz	2	5	*	*	*				
76	01110110	rorzx	2	6	*	*	*				
6E	01101110	rora	3	6	*	*	*				
7E	01111110	rorax	3	7	*	*	*				
40	01000000	rti	1	6	de la pile						CC <- pile, CO <- pile
60	01100000	rts	1	6							CO <- pile, ((CO)+1) -> CO
E9	11101001	sbc#	2	2	*	*	*			*	(A) - opérande - NON(C) -> A
E5	11100101	sbcz	2	3	*	*	*			*	
F5	11110101	sbczx	2	4	*	*	*			*	
ED	11101101	sbca	3	4	*	*	*			*	
FD	11111101	sbcax	3	4x	*	*	*			*	
F9	11111001	sbcay	3	4x	*	*	*			*	
E1	11100001	sbcx*	2	6	*	*	*			*	
F1	11110001	sbc*y	2	5x	*	*	*			*	
38	00111000	sec	1	2			1				1 -> C
F8	11111000	sed	1	2					1		1 -> D
78	01111000	sei	1	2				1			1 -> I
85	10000101	staz	2	3							(A) -> opérande
95	10010101	stazx	2	4							
8D	10001101	staa	3	4							
9D	10011101	staax	3	5							
99	10011001	staay	3	5							
81	10000001	stax*	2	6							
91	10010001	sta*y	2	6							
86	10000110	stxz	2	3							(X) -> opérande
96	10010110	stxzy	2	4							
8E	10001110	stxa	3	4							
84	10000100	styz	2	3							(Y) -> opérande
94	10010100	styzx	2	4							
8C	10001100	stya	3	4							
AA	10101010	tax	1	2	*	*					(A) -> X
A8	10101000	tay	1	2	*	*					(A) -> Y
BA	10111010	tsx	1	2	*	*					(S) -> X
8A	10001010	txa	1	2	*	*					(X) -> A
9A	10011010	txs	1	2							(X) -> S
98	10011000	tya	1	2	*	*					(Y) -> A

- **code**: le code opération, en hexadécimal dans les deux premières colonnes, en binaire dans les colonnes suivantes;
- **mnémonique**: le "nom" de l'instruction. Les mnémoniques sont composés de la façon suivante:

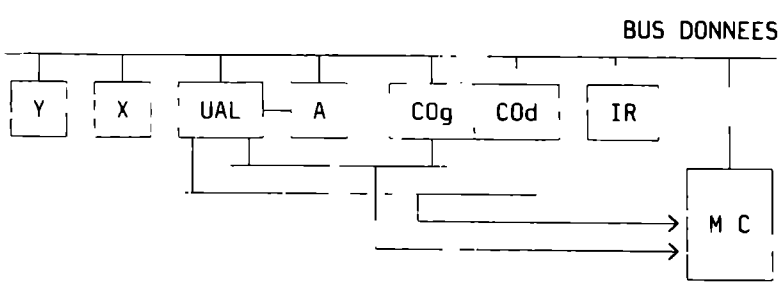
  * les trois premières lettres définissent l'**opération**,
  * les caractères suivants (éventuels) définissent le **mode** d'adressage:
    - . # adressage immédiat;
    - . z adressage page zéro;
    - . zx adressage page zéro indexé par X;
    - . zy adressage page zéro indexé par Y (seules LDX et STX en bénéficient);
    - . a adressage absolu (JMP et JSR en disposent sans que l'on n'ait ajouté "a" à leur mnémonique: il ne peut y avoir d'ambiguïté);
    - . ax adressage absolu indexé par X:
    - . ay adressage absolu indexé par Y;
    - . x* adressage indexé par X puis indirection (pré-indexation);
    - . *y adressage indirect puis indexation par Y (post-indexation);
    - . * indirect (seul JMP en est doté);
    - . implicite (cas de CLC par exemple);
      relatif à CO (cas des branchements Bxx);

- **lg.**: le nombre d'octets occupés par l'instruction;

- **cyc.**: le nombre de cycles pris par l'exécution de l'instruction:
  x : ajouter un cycle s'il y a changement de page,
  @ : ajouter un cycle si le branchement se fait dans la même page, deux cycles sil y a changement de page;

- **indicateurs**: les indicateurs sont modifiés par l'instruction (*), mis à 1 ou à 0. S'ils ne sont pas modifiés,la colonne correspondante contient un espace.

STRUCTURE DU 6502:



annexe B

références

ADA84 Modeling algorithm executive time on processor arrays.
L.M. Adams, T.W. Crockett.
Computer, juillet 84, page 38.

AGE82 Data flow systems.
Tilak Agerwala, Arvind.
Computer février 82, page 10.

ALE78 Bit sliced microprocessor architecture.
N.A. Alexandridis.
Computer, juin 1978, vol 11 numéro 6.

ALL84 A new series of high-performance real-time computers.
M.E. Allan, N. Schoendorf, C.B. Chatterton, D.M. Cross.
HP Journal, fév. 84, p. 3.

AME85 Simplicity in a microcoded computer architecture.
F.C. Amerson.
Hewlett-packard Journal, sept 85, page 7.

ARC85 The Datamation 100.
P. Archibold, J. Verity.
Datamation 1-er juin 85, page 37.

AUG84 Etude comparative de réseaux d'interconnexion dans une architecture MIMD.
M. Auguin, F. Boeri (LASSY).
Congrès sur de nouvelles architectures pour les communications, sept 84, Eyrolles.

AUT79 Cours d'assembleur VAX.
Autin.
Interne IIE.

AYA81 le plus gros des micro-processeurs.
JM Ayache, C. Beounes.
La Recherche, octobre 81, page 1144.

BAC81 L'AM29116 : un microprocesseur de 16 bits bipolaires pour les applications de contrôle.
M. Baconnier.
Minis et Micros numéro 146, page 27.

BAE80 Computer Systems Architecture.
J.-L. Baer.
Pitman Publishing Limited, London WC2B5PB, UK ISBN 273 01474 9, 600 pages.

BAL82 The NS16000 family.
S. Bal, A. Kaminker, Y. Lavi, A. Menachem, Z. Soha.
Computer juin 82, page 59.

BAS85 RISC Architecture's simple instruction set boosts CAD/CAE power.
E.J. Basart (Ridge Computer Inc.).
Computer Technology review, spring 85, page 39.

BAS85A IBM's early computers.
Ch. Bashe, L. Johnson, J. Palmer, E. Pugh.
Computerworld Extra, 24 déc. p. 25.

BEL78 The evolution of DECsystem10.
C.G. Bell, A. Kotok, T.N. Hastings, R. Hill.
Communications of the ACM, janvier 78, page 44.

BEL81 Conception d'un séquenceur de microprocesseur.
C. Bellon et G. Saucier.
RAIRO numéro 2 vol 15, 1981, page 131.

BHA82 Architecture management for ensuring software capability in the VAX family computer.
Dileep Bhandarkar.
Computer, février 82, page 87.

BIR85 De l'architecture à jeu d'instructions réduit à l'architecture de grande précision.
J. Birnbaum, W. Worley (HP).
Le Monde Informatique, 9 sept. 85, pages 11-16.
Hewlett-Packard journal, août 1985, pages 2-8.

BON69 Description of the 7600 Computer System.
P. Bonseigneur.
Computer Group News, mai 1969.

BON83 Analysis of the M6809 instruction set.
Joel Bonly (Motorola).
AFIPS Conference Proceedings, vol 52 1983 NCC, pages 503-511.

BOR78 The evolution of the Speery Univac 1100 series.
B.R. Borgerson, M.L. Hanson, P.A. Hartley.
Communications of the ACM, janvier 78, page 25.

BRO83 Classifying categories and historical development of circuit switching topologies.
G. Broomell, J.R. Heath.
Computing Surveys, vol 15, numéro 2, juin 83, pages 95-133.

BRO84 Understanding execution behaviour of software systems.
J.C. Browne (Univ. of TEXAS).
Computer, juillet 85, page 83.

BUR72 Burroughs B6700 Information Processing Systems.
Reference Manual 1972.

BUR77 Comparing architectures.
W.E. Burr, W.R. Smith.
Datamation fév 77, page 48.

BUT83 Supermini versus mainframe.
G. Butcher (Norsk Data).
Systems International, juillet 83, page 33.

CAH82 Superordinateurs.
J.P. Cahier.
Le Monde Informatique, numéro 81, 13 déc 82, page 9.

CAH82A Systèmes redondants: vacciner l'informatique contre les pannes.
J.P. Cahier.
Le Monde Informatique, 20 déc 82, page 6.

CAR82 Avec sa famille NS16000, National Semiconductor estime acquérir 21% du marché des 16/32 bits en 1985.
R. Carrasco.
Minis et Micros, numéro 167, page 21.

CAR83 8- and 16-bit processor family keeps pace with fast RAMs.
W. Carter, J. Hu, F. Lynch, D. Stevenson.
Electronic design, 28 avril 83 page 215.

CAR85 RISC-Y Business?
R.E. Carlyle.
Datamation, 1985, page 30.

CAR85A CAFS and relational processing.
J.W.S. Carmichael (ICL).
Convention Informatique 16-20 sept 85, page 45 tome B.

CAS78 Architecture of the System 370.
R.P. Case, A. Padegs.
Communications of the ACM, janvier 78, page 73.

CAS78A Array processors.
R.A. Caspe.
Mini-Micro Systems juillet 78.

CAS85 Der "Reduced-Instruction-Set-Computer" (RISC).
Brian Case.
Elektronik, 15 nov. 85, pages 61-68.

CHA81 An approach to scientific array processing: the architecture design of the AP-120B/FPS164 family.
A.E. Charlesworth.
Computer, septembre 81, page 18.

CHA85 La conception d'un micro-ordinateur strictement compatible IBM: l'exemple du Bull Micral 30.
B. Challet.
Convention Informatique 16-20 sept 85, page 214 tome A.

CHU81 Programming language and direct-execution computer architectures.
Y. Chu, M. Abrams.
Computer, juillet 82, page 22.

CIN82 L'avénement des 32 bits.
Fr. Cinare.
Minis et Micros numéro 157, page 35.

CIN82A Norsk Data : gamme et performances accrues pour la 3-ème génération.
F. Cinare.
Minis et Micros numéro 160, page 85.

CLA85 Moving from the 8086 to the 80286.
W.J. Claff.
Byte, fall 85, p. 93-101.

COH82 L'IAPX286 d'INTEL: pas si micro que ça.
J. Cohen.
Le Monde Informatique, 8 février 82, page 7.

COM83 Les supercalculateurs.
D. Comte, JC Syre.
La Recherche, septembre 83, page 1084(14).

CON79 The IBM 3033: an inside look.
W.D. Connors, J.H. Florkowski, S.K. Patton.
Datamation1979, page 198.

CON81 A l'intérieur du processeur IBM 3033.
W.D. Connors, J.H. Florkowski, S.K. Patton.
Informatique nouvelle janvier 81, page 27.

CON85 What's a 'microsuper'?
J. Connolly.
Computerworld 21 oct 85,pages 39,44.

COU82 Une cinquième génération d'ordinateurs.
G. Couturier.
La Recherche, février 82, page 230.

CRA80 The elements of single-chip microcomputer architecture.
H. G. Cragon (Texas Instr).
Computer, octobre 80, page 27.

CW80 IBM double header gives first glimpse of 'H'.
Computerworld 17 nov 80, page 1.

CW81 Harris adds mid-range 48-bit mini.
Computerworld, 6 avril 81, page 89.

CW81A Superminis seen topping $300 Million in '81.
Special report.
Computerworld, 27 juillet 81.

CW82 CSP unveils high-speed array processor.
Computerworld 1-er mars 82, page 62.

CW82A Superminis offer mainframe power, mini costs.
Computerworld, 1-er mars 82, page 10.

DAS80 Some aspects of high level microprogramming.
S. Dasgupta.
ACM Computing surveys, septembre 1980, page 295.

DAT82 Reviewing Europ's top 25.
Datamation novembre 82, page 124.

DAV80 Ballistic missile defense: a supercomputer challenge.
Carel G. Davis, R. L. Couch.
Computer nov 80, page 37.

DEC75 LSI/11 PDP11/03 processor handbook.
DEC 1975.

DEC78 VAX11 software handbook.
DEC 1978.

DEC79 VAX11 architecture handbook.
DEC 1979.

DEC79A VAX11/780 hardware handbook.
DEC 1979.

DEH85 Liaison des VLSI avec le monde extérieur.
G. Dehaine, K. Kurzweil (Bull).
Convention Informatique 16-20 sept 85, tome B, pages 26-29.

DEN80 Data flow supercomputers.
Jack B. Dennis.
Computer nov 80, page 48.

DID85 How compatible is it?
S. DiDonato (Data Decisions).
Computerworld, In Depth, 17 juin 85, pages 43-53.

DUB85 Panorama des microprocesseurs 32 bits.
R. Dubois (Cegos).
Convention Informatique 16-20 sept 85, page 254 tome A.

EI81 NS, Eurotechnique et Fairchild s'apprêtent à lancer le dernier microprocesseur 16 bits: le 16000.
Electronique Industrielle, mai 81, page 9.

ENS77 Multiprocessor organization: a survey.
P.H. Enslow.
ACM Computing Survey, mars 1977.

EWA84 Recent benchmarks of high performance computers.
R.H. Ewald (Cray Research).
SEAS AM84, page 638.

FAL69 Calculatrice électronique.
J. Falmagne, P.E. Piché.
La science pour tous, 1969, tome 8, page 147.

FAR Comparing the 4341 and M80/42.
Dale F. Farmer.
Computerworld (in depth).

FAR81 IBM-compatible giants.
Dale F. Farmer.
Datamation déc 81, page 92.

FER85 Nouvelle concurrence dans la FT. Etat des lieux.
Fertig.
Ordinateurs, 20 mai 85, page 34.

FIS82 Les fournisseurs compatibles en France.
K. Fisher.
01 Hebdo numéro 726, 6 déc 82, page 7.

FLO Cours d'architecture des ordinateurs.
Manuel d'enseignement de l'IIE.
Florin, E. Pichat.

FLY78 Computer organization and architecture.
M.J. Flynn.
dans Operating Systems chez Springer Verlag.

FOT84 An adaptable 1-MIPS real-time computer.
D.A. Fotland, L.S. Moncton, L.E. Neft.
HP Journal fév. 84, p. 7.

FRA76 The second half of the computer age.
W.L. Franck.
Datamation, mai 76, page 91.

FRE76 IBM and multiprogramming.
D.N. Freeman.
Datamation, mars 76, page 92.

FRE85 Coprocessor chips unlock throughput of 68000 family.
M. Freeman, C. Kaplinsky (Signetics).
Computer Technology Review, spring 85, p. 33-43.

FRI85 The 8087/80287 performance curve.
S.S. Fried.
Byte, fall 85, p. 67-88.

GAJ82 A second opinion on data flow machines.
D.D. Gajski, D.A. Padua, D.J. Kuck, R.H. Kuhn.
Computer, février 82.

GAL85 Des ordinateurs de plus en plus puissants pour des calculs de plus en plus rapides.
Ch. Galus.
Le Monde, 4 sept 85, page 19.

GEI81 Design approach to 16 bits.
P. Geiler (Zilog).
Systems, nov 81, page 24.

GIL82 Introduction aux microprocesseurs.
C.M. Gilmore.
MacGraw Hill.

GOR85 A walk through the Computer Museum with Gordon Bell.
Gordon Bell.
Computerworld 14 oct 85, In Depth, pages ID/10-23.

GOU80 Naissance et évolution de l'industrie informatique.
VI. L'architecture.
P. Goujon.
Micro-Systèmes, sep-oct 80, page 60.

GRA85 Multiuser micros add value to PC LANs.
F.T. Granville.
Mini-Micro Systems, août 85, p. 55.

GRE85 Et maintenant la course aux ordinateurs géants.
G Gregory.
L'usine nouvelle, numéro 17, 25 avril 1985, pages 60-67.

HAB85 Multiprocessor Technology means more muscle, less fat.
L. Haber.
Mini-Micro Systems, juin 85, p. 83.

HAN82 Optimization of the MIPS rate by memory mapping.
A. Hanzal (UBS).
ECOMA10 proceedings oct 82, page 361.

HAY82 A survey of highly parallel computing.
L.S. Haynes, R.L. Lau, D.P. Siewiorek, D.W. Mizell.
Computer, janv 82, page 9.

HAY85 Digital Equipment Corp. Family tree.
M.J. Hayes.
Computerworld, 14 oct 85, page 95

HAY85A IBM family tree.
M.J. Hayes.
Computerworld Extra, 24 déc. 85. p. 22.

HEN Inside the IBM System/38.
G.G. Henry, R.L. Hoffman, F.G. Soltis, C.T. Watson, G.F. Aberle, F.E. Benson, P.T. Taylot.
Computerworld, In Depth.

HEN81 The Cray-1/S, the Cyber 205.
Tom Henkel.
Computerworld 9 nov 81, page 17.

HEN81A Hardware roundup: a look at 98 systems from 17 vendors.
Tom Henkel.
Computerworld 13 juillet 81, page 10.

HEN82 Hardware round up: a look at 147 systems from 26 vendors.
T. Henkel.
Computerworld, 2 août 82, page 23.

HEN83 Hardware roundup.
T. Henkel.
Computerworld, 8 (centraux et superminis), 15 (minis), 22 (micros) août 83.

HEN85 Hardware roundup.
T. Henkel.
Computerworld 19 août 85 (centraux), 26 août 85 (super), 2 sept 85 (micros et minis).

HEN85A Sperry was an early bird at dawn of computing.
T. Henkel.
Computerworld, 18 fév. 85, p. 65.

HEY83 The 8086 - An architecture for the future.
S.A. Heywood.
Byte, juin 83, page 450.

HIG83 A vector processing tutorial.
Lee Higbie.
Datamation, août 83, page 120.

HOB74 Minicomputer survey.
L.C. Hobbs, R.A. McLaughlin.
Datamation juillet 74, page 50.

HOC84 Novel computer architectures.
R.W. Hockney (Reading Univ.).
SEAS AM84, page 621.

HOG77 The past and thoughts about semiconductor technology.
C.L. Hogan.
Interface Age, mars 77.

HOL79 VM370 Asymmetric Multiprocessing.
L.-H. Holley, R.P. Parmelee, C.A. Salisbury, D.N. Saul.
IBM Systems, numéro 1, 1979, p. 47.

HON76 Level 6 minicomputer handbook.
AS22, Rev.0, janvier 76.

HOW85 The FT crowd.
Ch. Howe.
Datamation mai 85, pages 63-65.

HOW85A Back to the basics.
C.L. Howe. Datamation, 1985, page 54-60.

HUS82 Single Chip microcomputers can be easy to program.
Bill Huston.
AFIPS conference, proceedings vol 51, 1982 NCC page 85.

IBM System/370 principles of operation.
GA22-7000-3.

IBM83F System 370 Extended Architecture. Principles of Operation.
IBM SA22-7085-0 mars 83.

IBM84 IBM 3083 functional characteristics.
IBM GA22-7083-3, mai 84.

IBM85 IBM System/370. System summary: processors.
IBM GA22-7001-15, février 85.

INM85 Define, the tangled web of micros, minis, mainframes.
W. Inmon.
Computerworld 29 juillet 85, page 65,66,70.

INT81 Introduction to the IAPX432 architecture.
INTEL, 171821-001, 1981.

INT85A iAPX 286 high performance benchmark report.
INTEL, 231558-001, mai 85, 51 pages.

INT85B iAPX 286/10 high performance microprocessor.
INTEL 210253-008, mai 85, 55 pages.

ISA82 Squeezing the most out of the 68000.
J. Isaak (Charles River Data Systems).
Mini-Micro Systems, oct 82, page 193.

JAU80 La micro-programmation des microprocesseurs.
P. Jaulent.
Micro-Systèmes, sep-oct 80, page 125.

JAU81 Le micro-processeur et son environnement.
P. Jaulent.
Micro-Systèmes, mars 81, page 117.

KAR81 Architectural and software issues in the design and application of peripheral array processors.
W.J. Karplus, D. Cohen.
Computer, sept 81, page 11.

KAT78 8086 microcomputer bridges the gap between 8- and 16-bit designs.
B.J. Katz, S.P. Morse, W.B. Pohlman, B.W. Revenel (Intel).
Electronics fév. 78, p. 99.

KEH81 Battle of the 16 bit micros.
L. Kehoe.
Systems, sept 81, page 17.

KER85 Making micro money via vertical market.
S.B. Kern.
Datamation, 15 avril 85, p. 84-8.

KEV79 New options from big chips (8086).
J. Mc.Kewitt, J. Bayliss.
IEEE Spectrum, mars 79.

KLE85 Distributed systems.
L. Kleinrock.
Communications of the ACM, nov 85, vol 28 num. 11 p. 2000.

KOS80 Second generation of vector supercomputers.
E.W. Kosdrowicki, D.J. Theis.
Computer novembre 80, page 71.

KOZ83 Supercomputers for the eighties.
E.W. Kozdrowicki.
Digital Design mai 83, page 94.

KUN82 Why systolic architectures.
H.T. Kung.
Computer, janv 82, page 37.

KUR82 What to expect from the 5-th generation computer.
Shohei Kurita.
Computerworld, 1982, In Depth.

LAM Minis and Micros stealing the show?
FR. Lamond.
Datamation, page 192-2.

LAN80 Local microcode compaction techniques.
D. Landskov, S. Davidson, B. Shriver, P.W. Mallett.
ACM Computing surveys, septembre 1980, page 261.

LAR84 Multitasking on the CRAY X-MP-2 multiprocesseur.
J.M. Larson (Cray).
Computer, juillet 84, page 62.

LAR84A Un conte moderne: l'histoire édifiante des relations entre les nombres flottants et les ordinateurs.
Ph. Larcher.
Minis et Micros num. 227 p. 49.

LAT83 Des géants pour comprendre l'univers.
J. Latour (CNRS).
Le Monde Informatique 31 janvier 83, page 8.

LAV78 The Manchester Mark I and Atlas: a historical prespective.
S.H. Lavington.
Communications of the ACM, janvier 78, page 4.

LAZ85 Les technologies de fabrication des circuits intégrés 1985-1990.
J. Lazzari (LETI).
Convention Informatique 16-20 sept 85, tome B, pages 4-10.

LEN84 Les réseaux d'interconnexion.
J. Lenfant, C. Michel.
Congrès sur de nouvelles architectures pour les communications, sept 84, Eyrolles.

LEV82 Virtual Memory management in the VAX/VMS operating system.
H.M. Levy, P.H. Lipman.
Computer, mars 82, page 35.

LIA80 Tracking the elusive KOPS.
E.J. Lias.
Datamation novembre 80, page 99.

LIL81 Dossier micro-processeurs: 35 fabricants pour 100 familles ou types de produits.
H. Lilen.
Electronique Industrielle, mai 81, page 29.

LIP78 Developments of directions in Computer Architecture.
G.J. Lipovski, K.L. Doty.
Computer, août 78.

LIT83 Texas Instruments' 99/2 Basic Computer.
H. Littlejohn, M. Jander.
Byte, juin 83, page 128.

LMI86 750 000 ordinateurs dans l'hexagone.
Le Monde Informatique, 13 janvier 86, p. 34.

LON82 Reliability of computer systems.
R. Longbottom.
ECOMA10 proceedings oct 82, page 260.

LOR84 Panorama des circuits et coprocesseurs pour calcul arithmétique.
Ph. Lorrain.
Minis et Micros numéro 216, pages 107-111.

LR80 De l'ordinateur classique au supercalculateur.
La Recherche numéro 110, avril 80, page 456.

LUN77 Empirical evaluation of some features of Instruction set processor architectures.
A. Lunde.
Communications of the ACM, mars 77, page 143.

MAL82 Sortir des machines de Von Neumann pour sortir de la crise du logiciel.
Rex Malik.
01 hebdo numéro 698, 24 mai 82, page 38.

MAR78 Les 'jumbo' ordinateurs.
Cl. Marson.
01 informatique numéro 125, novembre 78, page 44.

MAR81 Memory interference models for a multi-micro-processor system with a shared bus and a single external common memory.
M.A. Marsan, F. Gregoretti.
Euromicro journal, février 81, page 124.

MAR82 Les découvertes d'IBM.
Y. Marcel.
L'informatique professionnelle numéro 3, mai 82, page 35.

MAR85 Les différentes formes et les différents niveaux de compatibilité.
C. Marson (ELF).
Convention Informatique 16-20 sept 85, page 210 tome A.

MAS82 Talks about the fifth-generation computer.
Yoneji Masuda.
Computerworld, 1982, In depth.

MDO84 Instruction-Level program and processor modeling.
M.H. MacDougall (Amdahl).
Computer juillet 84, page 14.

MEA83 Introduction aux systèmes VLSI.
C. Mead, L. Conway.
Inter Editions 400 pages.

MEL83 Digital's Professional 300 series : a minicomputer goes micro.
W. Melling.
Byte, juin 83, page 96.

MIC80 Dix micro-processeurs 8 bits.
Micro-Systèmes, sept-oct 80, page 67.

MIC81 Une introduction aux micro-processeurs.
Micro-Systèmes, janv-fév 81, page 100.

MIC81A Présentation du NSC800.
Micro-Systèmes, janvier 81, page 96.

MIC81B Les micro-processeurs 16 bits.
Micro-Systèmes, mars 81, page 69.

MIC81C IAPX432: un microprocesseur 32 bits.
Micro-Systèmes, mai 81, page 76.

MIT84 La conception d'une unité centrale microprogrammable.
D. Mithani (AMD).
Electronique, Techniques et Industrie, 1984, numéro 8, pages 29-36.

MOD78 Reference Manual, Modcomp central processor 210-140000-000.
Modcomp.

MON83 Tight squeeze : the HP series 200 model 16.
J. Monahan.
Byte, juin 83, page 110.

MOR78 The Intel 8086 microprocessor : a 16 bit evolution of the 8080.
S.P. Morse,W.B. Pohlman, B.W. Ravenel (Intel Corp.).
Computer, juin 1978, vol 11 numéro 6.

MOR80 Intel microprocessors: 8008 to 8086.
S.P. Morse, B.W. Ravenel, S. Mazor, W.B. Pohlman (Intel Corp).
Computer octobre 80, page 43.

MOR81A Des "benchmarks" inédits sur le bas de gamme PDP11.
F. Morini.
01 Hebdo numéro 647, 1-er juin 81, page 8.

NAT81 NS16032 high performance microprocessor.
National Semiconductor, août 81.

NAU85 Die M68000-Familie.
A. Nausch.
Elektronik, 15.11.85, page 115.

OLI83 CPS/32 Systeme (Stratus 32).
Olivetti GS code 3932880G (0), juin 82.

ORD82 Le 3081-D sur la sellette.
Ordinateurs 1-er mars 82, page 11.

ORD85 XA: maintenant il faut y aller!
Ordinateurs, 20 mai 85, page 10.

ORG78 Interpreting machines: architecture and programming of the B1700/B1800 series.
E.I. Organick, JA Hinds.
North-Holland.

OUA83 Successeur du Z80, le Z8108 est le premier membre de la famille Z8000.
M. Ouakine.
Minis et Micros, numéro 182, page 35.

PAL80 The Intel 8087 numeric data processor.
J. Palmer.
7th annual symposium on computer architecture, 1980, page 174.

PEU79 Architecture of a new microprocessor: Z8000.
B.L. Peuto (Zilog Inc).
Computer, février 1979, vol 12 numéro 2.

POH81 Computer Memory Systems.
A.V. Pohm, T.A. Smay.
Computer, oct 81, page 93.

POW83 The everlasting mainframes : how large computers survive DDP.
D.R. Powell.
Computerworld, In Depth, 27 juin 83.

PRI83 Les mini-ordinateurs 32 bits. II - Eclipse de Data General.
V. Prince.
Minis et Micros, numéro 195, page 59.

QUI85 Les hyper-ordinateurs.
P. Quinton.
La Recherche num. 167, juin 85 p. 740.

RAM74 Efficiency in generalized pipeline networks.
C.W. Ramamoorthy, H.F. Li.
NCC 1974, page 625.

REG85 Introduction du système parallèle 300 XR.
N. Reggaï.
Réponses numéro 8, sept. 1985, page 22.

RIC81 The chief architect's relexions on Symbol IIR.
Rex Rice.
Computer, juillet 81, page 49.

RIE74 Instruments systems are getting 'smarter'.
M.J. Riezenman.
Electronics, 11 juillet 74.

ROC81 Supercomputers: only a few success stories.
Jack Rochester.
Computerworld 9 novembre 81, page 18.

ROG76 Un recueil comparatif des ordinateurs: Procedata.
A. Rodriguez.
01 Informatique août/sept. 76, page 11.

RUS78 The Cray-1 computer system.
R.M. Russel.
Communication of the ACM, janvier 78, page 63.

SAT79 Design trade-offs in VAX-11 translation buffer organization.
M. Satyanarayanan, D. Bhandarkar.
Computer, déc 81, page 103.

SAT80 Commercial multiprocessing systems.
M. Satyanarayanan.
Computer, mai 80, page 75.

SCA82 PE 3200s gain Fortran optimizing compiler.
T. Scannell.
Computerworld, 22 février 82, page 6.

SCH80 Experience using multiprocessor systems - a status report.
P. Schwarz, A.K. Jones.
ACM Computing surveys, juin 80, page 121.

SER84 Fault-tolerant systems in commercial applications.
O. Serlin (ITOM Int. Company).
Computer août 84, page 19.

SEI85 Pyramid challenges DEC with RISC supermini.
M. Seither.
Mini-Micro Systems août 85, p. 33-35.

SER85 Parallel processing: fact or fancy?
O. Serlin.
Datamation, 1985, page 93.

SHI78 Two versions of 16-bit chip span micro-processor, minicomputer needs.
M. Shima (Zilog Inc.).
Electronics, 21 décembre 1978, page 81.

SHI85 80386 cache design.
Glen Shires (Intel).
Solutions, nov/déc 85, pages 22-27.

SHO83 Microcoprocessor chip integrates a host of peripheral functions to simplify systems.
K. Shoemaker.
Electronics, 5 mai 83, page 139.

SIE79 Interconnections networks for SIMD machines.
H.J.Siegel.
Computer, juin 79, page 57.

SIE81 The multi-stage cube: a versatile interconnexion network.
H.G. Siegel, R.M. McMillen.
Computer, déc 81, page 65.

SIE84 Architecture of fault-tolerant computers.
DP.P Siewiorek (Carnegie-Mellon univ.).
Computer août 84 page 9.

SMI AADC computer family architecture questions and answers.
W.R. Smith, B. Wald.

SMI77 Multiprocessor Memory Organization and Memory Interference.
A.J. Smith.
Communications of the ACM, oct 77, page 754.

SMI82 Cache memories.
A.J. Smith.
ACM Computing surveys, septembre 82, page 473.

SRE80 An experimental study of relative throughput in a multiprocessor computer system.
K. Sreenivasan, GA Nelson, JA Maksin.
Software Practice and Experience, numéro 10, 1980, page 973.

STA83 Design Philosophy behind Motorola's MC68000.
T.W. Starnes.
Byte, juin 83, page 339.

STR79 A microprocessor architecture for a changing world : the Motorola 68000.
Ed. Stritter, T. Gunter (Motorola Inc).
Computer, février 1979.

SYS81 The 32 bit mini.
Systems, sept 81, page 53.

SYS81A Survey of 16 bit micros: micromovements.
Systems, oct 81, page 45.

TAG85 Speculations on the evolution of an architecture.
A.G. Tagg.
Computer Architecture News, vol. 13 numéro 2, juin 1985.

TAN76 Tandem 16 system description.
TANDEM, 11 octobre 1976.

TAV81 Le micro-processeur 6809.
C. Tavernier.
Micro-Systèmes, nov-déc 81, page 63.

THE71 Minis revisited.
D.J. Theis, L.C. Hobbs.
Datamation 15 mai 1971, page 25.

THO85 Le marché de la micro-informatique dans la dernière ligne droite.
C Thomas.
Convention Informatique 16-20 sept 85, page 43 tome A.

TRA83 Systems with staying power : a design approach to fault tolerance.
C.G. Trass, L.S. Bensky.
Computerworld, 14 février 83.

TRE82 Japan Fifth-generation computer systems.
PH. C. Treleaven, I.G. Lima.
Computer, août 82, page 79.

TR81 Le plus puissant ordinateur du monde: Cyber 205.
Temps Réel numéro 6, page 43.

UBE85 Silicon chip advances optimize integration of 32-bit microprocessor.
A. Uberoi (NSC).
Computer Technology Review, spring 85, page 17.

UHL81 The office of the future.
Uhlig, Farber, Bair.
North-Holland 0 444 85 336 7.

VER82 1982 Mini-Micro survey.
J.W. Verity.
Datamation, nov 82, page 34.

YUV76 Cray 1 for 6000/7000 hackers.
G. Yuval.
Software and practice.

YAS82 Tokyo looks to the '90s.
E. Yasaki.
Datamation janvier 82, page 110.

YAS82A Fail-safe vendors emerge.
E.K. Yasaki.
Datamation, nov 82, page 51.

annexe C

exemples de cycles et de MIPS

CONSTRUCTEUR	MODELE	CYCLE ns	PUISSANCE RELATIVE	MIPS
IBM	135 BC mod	275-1430		
	EC mod	330-1485		
IBM	115-0	480	2,3	
CAMBRIDGE	Model 1	480-1130	2,5	
IBM	125-2	480	6,3	
CAMBRIDGE	Model 2	300-950	6,9	
IBM	4321	900(1)	11	0,19
IBM	4311-1	900(1)	11	0,2
IBM	138	275-1430	11,7	
IBM	135-3	275-1485	11,7	
CAMBRIDGE	Model 3	180-850	12,9	
MAGNUSON	M80/30	100	16	0,24
CAMBEX	1636-1	50	26	0,39
NANODATA	VMX200	300	16	
NIXDORF	8890-30	200	15	0,25
NIXDORF	8890/CP	200	17	0,25
Magnuson	M80/3	100	17	
NAT.SEMI.C	400	100	18	
NAT. CSS	3200	250	18	
ITEL	AS/3-3	115	18	
IBM	4331-11	900(1)	18	0,26
CITEL	30 Model 3	200	21	
CDC	Omega480-1	50	22	0,38
IBM	4361-3	100	22	0,38
CAMBEX	1636	50	23	0,39
IBM	4331-2	900(1)	22	0,38
IBM	4341-9	150-300	24	0,4
IBM	4341-9	150-300	24	0,4
IBM	148	180-225	24	
ITEL	AS/3-4	115	24	
Magnuson	M80/31	100	26	0,42
IPL	4436	50	27	0,43
CDC	Omega480II	50	28	
ITEL	AS/4	115	28	
NIXDORF	8890-50	200	30	0,5
Magnuson	M80/32	100	32	0,56
GLOBAL U.S.	USX40	100	32	0,65
NANODATA	VMX400	175	32	
CAMBEX	1641	50	35	0,59
CAMBEX	1636-10	50	36	0,6
IBM	4341-10	150-300	34	0,58
CITEL	30 Model 4	200	36	
Magnuson	M80/4	100	36	
IBM	4341-1	150-300	40	0,72
NAS	3000N	115	40	0,72
CAMBEX	1641-1	50	40	0,72
NIXDORF	8890-70	200	41	0,7
ITEL	AS/5-1	115	41	
Magnuson	M80/42	100	45	0,77
IBM	158-3	115	45	
ITEL	AS/5-3	115	45	
IPL	4443	50	45	1
UNIVAC	90/80-3	98	46	0,78
IBM	4361-4	100	49	0,79
IBM	4341-11	120-240	50	0,88
NAS	3000	115	50	0,88
NAS	5000N	92	50	0,88
CAMBEX	1651	50	52	0,90
CAMBEX	1641-11	50	52	0,90
Magnuson	M80/43	100	54	0,92
ITEL	7031	100	54	
IPL	4445	50	56	0,94
IBM	3031	115	54	1,1
NAS	5000E	92	60	1,
CAMBEX	1651-1	50	60	1
NAS	5000	92	62	1,1
IBM	4341-2	120-240	66	1,1
IBM	4361-5	100	66	1,14
IPL	4446	50	70	1,5
NAS	6130	75	66	1,1
Magnuson	M80/44	100	67	
NAS	6620	60	74	1,6
IBM	4341-12	115-230	76	1,2
IPL	4460	50	80	1,6

CONSTRUCTEUR	MODELE	CYCLE ns	PUISSANCE RELATIVE	MIPS
ITEL	7031AP	100	85	
NAS	6150	60	86	1,4
NAS	7000N	72	105	1,8
IBM	3031-AP	58	90	1,9
NAS	6630	60	100	2
IBM	4381-1	68	100	2,1
NAS	6650	50	125	2,4
AMDAHL	470V/7C	29	132	2,7
IBM	3033S	57	132	2,3
IBM	3032	80	124	2,7
IBM	168-3	80	124	2,7
AMDAHL	470V/5	32,5	130	
IBM	4381-2	68	133	2,7
IPL	4480	50	136	2,7
ITEL	AS/6-1	72	137	
NAS	AS6600	43	140	2,8
NAS	7000	72	143	2,7
AMDAHL	470V/5-II	32,5	143	
IBM	3083CX	24	145	2,4
NAS	AS/8023	40	146	2,9
ITEL	AS/6-2	72	149	
AMDAHL	470V/7B	29	172	3,5
AMDAHL	470V/6	32,5	177	
IBM	3083-E	26	185	3,1
IBM	3083EX	24	197	3,3
AMDAHL	470V/6-II	32,5	200	
AMDAHL	470V/7A	29	220	4,5
IBM	3033-N	57	210	4,6
NAS	9000N	48	214	4,6
Siemens	7571	52		4,5
IBM	3033-U	57	237	5,0
IBM	4381-3	68	239	4,8
NAS	9040	38	326	7,2
NAS	7000DPC	72	243	5,4
IBM	3083-B	26	277	5,7
Amdahl	470V/7	29	269	5,5
IBM	3083BX	24	295	6
NAS	9050	38	408	9
Amdahl	470V/8	26	318	6,5
NAS	9000N	48	314	7,1
IBM	3033-AP	57	405	9
IBM	3083-J	26	370	7,9
NAS	6630	60		
NAS	6650	50		
BASF	7/60	85		
BASF	7/65	60		
BASF	7/68	50		
BASF	7/70N	72		
BASF	7/70	72		
Amdahl	5840	23,25	378	8,4
BASF	7/80	40		
Amdahl	5850	23,25	522	11,6
IBM	3033-MP	28	405	9
BASF	7/80	40		8 - 9
NAS	9000-2	38	393	8,3
IBM	3081-D	26	465	10
IBM	3081-G	26	500	11,4
NAS	9060	30	525	11,2
IBM	3081 GX	24	545	12,5
NAS	9070	38	754	16,2
AMDAHL	5860	23,25	636	14

CONSTRUCTEUR	MODELE	CYCLE ns	PUISSANCE RELATIVE	MIPS
IBM	3081-K	26	675	18,8
NAS	9000DPC	38	708	20,2
IBM	3081 KX	24	735	16,3
NAS	9080	30	975	20
AMDAHL	5870	23,25	1197	26,6
AMDAHL	5880	23,25	1197	26,6
IBM	3084	26	1282	27
IBM	3090 200	18,5	1323	29,3
IBM	3084 QX	24	1384	29,1
IBM	3090 400	18,5	2381	52,7
NAS	AS/XL-60		1312	28
NAS	AS/XL-80		2437	50
IBM	3090	18,5	1323-2381	29 - 52
AMDAHL	5890	15	1556-4321	27 - 75
IBM	S/38-3	1100(1)	7	0,11
IBM	S/38-4	600(1)	13	0,2
IBM	S/38-5	600(1)	16	0,24
IBM	S/38-6	400(1)	20	0,3
IBM	S/38-7	400(1)	32	0,52
IBM	S/38-8	400(1)	32	0,52
IBM	8130	1500	12	0,22
IBM	8140A et B	800	20	0,36
IBM	8140C	800		0,5
IBM	S1-1			0,11
IBM	S1-2			0,22
IBM	S1-3			0,33
IBM	S1-4			0,43
UNIVAC	80-3	180	12	0,2
UNIVAC	80-4	180	14	0,2
UNIVAC	80-5	180	18	0,26
UNIVAC	80-6	180	21	0,37
UNIVAC	1100/61-C1			0,65
UNIVAC	1100/61-H1	116	70	1,2
UNIVAC	1100/81	50	114	2
UNIVAC	1100/82	50	208	4,5
UNIVAC	1100/83	50	307	5,9
UNIVAC	1100/84	50	397	8,4
UNIVAC	1100/91	30	345	7,5
UNIVAC	1100/92	30	644	14
UNIVAC	1100/93	30	920	20
UNIVAC	1100/94	30	1196	265
BULL	DPS7/35	330	16	0,24
BULL	DPS7/45	330	22	0,38
BULL	DPS7/35E	330	28	0,5
BULL	DPS7/55	140	29	0,45
BULL	DPS7/65	140	40	0,72
BULL	DPS7/65E	140	76	1,36
BULL	DPS8/20	asynch	22	0,38
BULL	DPS8/44	"	35	0,59
BULL	DPS8/52	"	61	1
BULL	DPS8/50	"	66	1,1
BULL	DPS8/62	"	82	1,4
BULL	DPS8/70	"	57- 384	1,8-8,2
BULL	DPS88/41		219	4,8
BULL	DPS88/82		470	13
BULL	DPS90/91		493	10,8
BULL	DPS90/94		1676	36,7
BULL	DPS6/92	300	38	0,62
BULL	DPS6/85	300	73	1,2
BULL	DPS6/95	125	110	1,8
Burroughs	B1955 (2)	167ou 250	19	0,27

CONSTRUCTEUR	MODELE	CYCLE ns	PUISSANCE RELATIVE	MIPS
Burroughs	B2900		28	0,44
Burroughs	B5900		30	0,54
Burroughs	B5920		33	0,64
Burroughs	B6900		61	1
Burroughs	B4955		120	2,1
Burroughs	B7800		199	4,4
Burroughs	B7900K	125	1185	28
Burroughs	A15-N	67	2537	59,5
ICL	39-30/1	190	60	1,3
ICL	39-80/1	25	495	11
CDC	180/810	50	59	0,8
CDC	170/720	50	47	0,79
CDC	170/825	50	47	0,79
CDC	170/810	50	61	1,0
CDC	170/730	50	65	1,1
CDC	170/835	56	71	1,2
CDC	170/845			
CDC	170/855	64	88	1,5
CDC	170/830	50	97	1,6
CDC	170/865	25	132	2,3
CDC	170/875	25	137	2,4
CDC	170/740	25	155	2,7
CDC	170/835	56	230	3,8
CDC	170/750	25	274	5,7
CDC	170/760	25	368	7,8
CDC	170/845	16	506	8,5
CDC	176	27,5	614	17,7
CDC	170/855	16	743	12,5
CDC	170/990	16	1914	32,3
CDC	180/990	16	2488	42
PRIME	2250			0,4
PRIME	250II	160	32	0,56
PRIME	550II	160	38	0,62
PRIME	750	160	60	1,06
PRIME	850	160	93	1,63
PRIME	9955	80	229	4
DEC	VAX11/725	810	20	0,36
DEC	VAX11/730	810	20	0,36
DEC	VAX11/750	400	40	0,72
DEC	VAX11/780	290	62	1,06
DEC	VAX11/782	290	109	1,9
DEC	VAX11/785	166	105	1,7
DEC	VAX8600	80	260	4,4
DEC	1091	133	72	1,3
AT&T	3B20S	400	60	1
AT&T	3B20A	400	95	1,6
HARRIS	60	300	45	0,76
HARRIS	80	300	36	0,6
HARRIS	100	300	36	0,6
HARRIS	300	300	44	0,76
HARRIS	600	300	45	0,76
HARRIS	80	300	50	0,88

CONSTRUCTEUR	MODELE	CYCLE ns	PUISSANCE RELATIVE	MIPS
HP	3000/40	105	26	0,45
HP	3000/44	27	32	0,56
HP	3000/64	75	62	1,1
HP	3000/68	75	96	1,5
DG	C/350	700	24	0,4
DG	MV/4000	200	36	0,6
DG	MV/6000	220	26	0,45
DG	MV/8000	400	60	1,4
DG	MV/8000II	220	71	1,2
DG	MV/10000	140	138	2,5
PE	3210	250	50	0,88
PE	3220	250	50	0,88
PE	3230	250	50	0,88
PE	3240	250	92	1,64
PE	3250	250	233	4,1
NCR	V8535-II	112	8	0,18
NCR	V8455	112	10	0,18
NCR	V8545-II	84	12	0,2
NCR	V8555-II	56	17	0,25
NCR	V8565-II	56	21	0,37
NCR	V8555M	84	17	0,29
NCR	V8565M	56	24	0,4
NCR	V8575-II	56	32	0,56
NCR	V8585-II	56	40	0,72
NCR	V8595-II	56	48	0,8
NCR	V8585M	56	50	0,9
NCR	V8650	38	116	2,2
NCR	V8670	38	212	4,7
Computer De sign System	Advisor 32/60	125	198	4,2
	32/80	125	475	14,6
WANG	VS90	160	45	0,77
WANG	VS100	160	67	1,3
WANG	VS300	120	201	3,3
STRATUS	32	125	48	0,72
CHARLES	68/10-80	500	23	0,42
RIVER DATA	UNIVERSE68	1000	33	0,57
	32/115		154	2,7
APOLLO	DN400	400	40	0,72
MICRODATA	SEQUEL	150	30	0,54
GOULD-SEL	32/8705	75	144	2,6
GOULD-SEL	32/8750	75	144	2,6
GOULD-SEL	32/9705			4,7
GOULD-SEL	32/8785	75	259	5,4
GOULD-SEL	32/9780			8,4
FORMATION	F4000/100	800	12	0,2
FORMATION	F4000/200	800	14	0,32
FORMATION	" AP	800	20	0,36
FORMATION	F4000/300	800	16	0,24
FORMATION	" AP	800	22	0,38

annexe D

liste des abréviations

AL	Adresse Logique
AM	AntéMémoire
ANSI	American National Standard Institute
AP	Adresse Physique *ou* Attached Processor
APL	A Programming Language
ASCII	American Standard Code for Information Interchange
AT&T	American Telegraph and Telephone
BASIC	Beginner's All-purpose Symbolic Instruction Code
BCD	Binary Coded Decimal
BL	Bus Local
BM	Bloc Mémoire
CA	Chiffre d'Affaire
CAO	Conception Assistée par Ordinateur
CB	Coupleur Bus
CD	Contrôleur de Disques
CDC	Control Data Corporation
CdD	Cadencement des Données
CI	Circuit Intégré
CII	Compagnie Internationale pour l'Informatique
CISC	Complex Instruction Set Computer
CM	Contrôleur Mémoire
CO	Compteur Ordinal
COBOL	COmmon Business Oriented Language
CP/M	Control Program for Microprocessors
CPU	Central Processing Unit
CRT	Cathode Ray Tube
CRTC	CRT Controller
CS	Control Store *ou* Code Segment sur 8086
DCB	Décimal Codé Binaire
DG	Data General
DEC	Digital Equipment Corporation
DOS	Disk Operating System
DPS	Distributed Processing System
DRAM	Dynamic RAM
EBCDIC	Extended Binary Coded Decimal Interchange Code
ECC	Error Correcting Code
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable PROM
EIS	Extended Instruction Set
ER	Executive Request
FORTRAN	FORmula TRANslator
FPA	Floating Point Accelerator
GCOS	General Comprehensive Operating System
GE	General Electric
GFLOPS	Giga opérations FLOttantes Par Seconde
HLL	High Level Language
HP	Hewlett-Packard
IA	Intelligence Artificielle
IBM	International Business Machines
IC	Integrated Circuit
IDC	International Data Corp.
IR	Indicator Register
KIPS	Kilo Instruction Par Seconde
KOPS	Kilo Opération Par Seconde
ISO	International Standardisation Organisation
LHN	Langage de Haut Niveau
LISP	LISt Processing

LSI	Large Scale Integration
MC	Mémoire Centrale
MCd	Mémoire de Commande
MEM	MEmoire Morte
MEV	MEmoire Vive
MFLOPS	Millions d'opérations FLOttantes Par Seconde
MHz	MégaHerz
MIMD	flot Multiple d'Instructions, flot Multiple de Données
MIPS	Millions d'Instructions Par Seconde
MIUD	flot Multiple d'Instructions, flot Unique de Données
MME	Master Mode Entry
MMU	Memory Management Unit
MOS	Metal-Oxyde Semiconductor
MP	MultiProcessor *ou* Mémoire Périphérique
MS-DOS	MicroSoft Disk Operating System
MSI	Middle Scale Integration
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
MV	série d'ordinateurs 32 bits de DG *ou* Mémoire Virtuelle
MVS	Multiple Virtual Storage
NCR	National Cash Register
NOVRAM	NOn Volatile SRAM
OA	Office Automation
OEM	Original Equipment Manufacturer
OI	Ordinateur Individuel
OSI	Organisation de Standardisation Internationale *ou* Open System Interconnection
PC	Processeur Central *ou* Personal Computer *ou* Program Counter
PES	Processeur d'Entrées/Sorties
PP	Peripheral Processor
PROM	Programmable ROM
RA	Registre Accumulateur
RB	Registre de Base
RG	Registre Général
RAM	Random Access Memory *ou* Registre Adresse Mémoire
RI	Registre Instruction
RISC	Reduced Instruction Set Computer
RL	Registre Limite (ou Longueur)
RM	Registre tampon Mémoire
ROM	Read Only Memory
RPG	Report Program Generator
RT	Registre de Translation
RX	Registre indeX
SBC	Single Board Computer
SIMD	Single Instruction flow, multiple Data flow
SISD	Single Instruction flow, Single Data flow
SRAM	Static RAM
SSC	Sous-Système Central
SSI	Small Scale Integration
SSP	Sous-Système Périphérique
SVC	SuperVisor Call
UC	Unité Centrale
UD	Unité de Disque
UE	Unité d'Exécution
UI	Unité de préparation des Instructions
UIMD	flot Unique d'Instructions, flot Multiple de Données
UIUD	flot Unique d'Instructions, flot Unique de Données
ULSI	Ultra Large Scale Integration
UT	Unité de Traitement
VAD	Value Added Dealer
VAR	Value Added Remaker
VLSI	Very Large Scale Integration
WCS	Witable Control Store
WDCS	Writable Diagnostic Control Store
WFI	Wafer Scale Integration
XA	eXtended Architecture
XDS	Xerox Data Systems

annexe E

index

MASSON, Editeur.
120, boulevard Saint-Germain
75280 Paris Cedex 06
Dépôt légal : Juillet 1986

IMPRIMERIE LOUIS-JEAN
av. d'Embrun, 05002-GAP
Dépôt légal : 195-mai 1986

Cette série d'ouvrages se propose de décrire **l'architecture des systèmes informatiques** tant matérielle que logicielle, qu'elle soit centralisée ou répartie.

Le domaine est très vaste. Les différents tomes dégagent les concepts utilisés par les ordinateurs et les illustrent de façon aussi complète que possible.

Tome 1 : Sous-système central
Tome 2 : Systèmes d'exploitation
Tome 3 : Entrées-sorties
Tome 4 : Périphériques
Tome 5 : Développement de programme

Cette série intéresse :

— *les enseignants et les étudiants,*
— *les professionnels de l'informatique voulant élargir ou approfondir leur domaine de compétence,*
— *les esprits curieux qui souhaitent comprendre les ordinateurs à l'aide d'une approche structurée et mieux les utiliser,*
— *tous ceux qui veulent suivrent l'évolution des systèmes informatiques.*

Le tome 1 parcourt **le processeur central** en partant du modèle purement séquentiel pour aboutir au processeur complexe faisant appel à tous les mécanismes de parallélisme et d'anticipation. Il différencie clairement deux grands niveaux d'architecture, interne et externe, en décrivant les objets physiques (registres, bus, mémoires, opérateurs, antémémoires, tampons, blocs) et logiques (instructions, données, sémaphores, processus...) qui les composent, ainsi que leurs relations.

9 782225 809156

ISBN:2-225-8

P7-DBE-358