

COLLECTION ARCHÉOLOGIE INFORMATIQUE

Une histoire de l'informatique

Digital Equipment Corporation

La saga d'une autre informatique

De Maynard à Compaq, 1957–1998

Sébastien Inion

Première édition — 2026

Mentions légales

Digital Equipment Corporation — La saga d'une autre informatique

© 2026 Sébastien Inion

Édition numérique — version 2.0

Publication de cette version : 10 Aout 2026

Cette édition numérique PDF est mise à disposition selon les termes de la licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International (CC BY-NC-ND 4.0).

Texte de la licence :

creativecommons.org/licenses/by-nc-nd/4.0/deed.fr

Cette licence ne couvre pas les photographies, reproductions de documents, marques et autres éléments provenant de tiers. Ceux-ci restent soumis à leurs droits et licences respectifs.

L'auteur conserve l'intégralité de ses droits d'exploitation commerciale, notamment pour la publication et la vente de l'édition imprimée.

Collection « Archéologie informatique »

Une collection du site *Histoire de l'informatique*

www.histoireinformatique.info

Table des matières

Avant-propos	ix
Prologue — Le monde d’IBM	1
I Deux ingénieurs face aux grands systèmes	7
1 Deux chemins vers Lincoln	9
1.1 Ken Olsen, le goût du concret	9
1.2 Harlan Anderson, de la physique à l’informatique .	11
1.3 Un rendez-vous dans l’histoire	13
2 Le laboratoire des possibles	15
2.1 Du calcul à l’interaction	15
2.2 La leçon de la modularité	17
2.3 Quitter une institution puissante	19
3 La naissance de DEC	21
3.1 Un projet prudent	21
3.2 L’ancienne filature	23
3.3 Les premiers pas	25
4 Commencer par les modules	29
4.1 Des briques pour les ingénieurs	29
4.2 Une documentation qui enseigne	33
4.3 Apprendre la série	34
4.4 La tentation de la machine complète	35

II	Rapprocher la machine	39
5	Le PDP-1, vendre une idée	41
5.1	La preuve par la démonstration	43
5.2	BBN, le premier client	44
5.3	Une autre manière de compter	46
6	Le PDP-1 au MIT	47
6.1	Pourquoi offrir une machine rare ?	48
6.2	Une machine presque nue	49
6.3	La rencontre avec les hackers	50
6.4	Un week-end pour fabriquer un outil	51
6.5	<i>Spacewar!</i>	53
6.6	Partager sans perdre l'interaction	55
6.7	Ce que le MIT rend à DEC	55
7	Le PDP-8 : douze bits pour changer d'échelle	57
7.1	Le PDP-5 ouvre la voie	57
7.2	Une architecture dictée par le prix	58
7.3	Le PDP-8 change d'échelle	62
7.4	Un ordinateur parfois invisible	65
7.5	Une architecture qui traverse les générations	66
7.6	Le mini-ordinateur devient un marché	67
III	Du laboratoire au marché	69
8	Brancher le monde réel	71
8.1	Le télétype, porte d'entrée ordinaire	72
8.2	DECtape	74
8.3	Du disque au système de stockage	78
8.4	Voir, mesurer, agir	79
9	Partager la puissance	83

9.1	Une grande machine selon DEC	84
9.2	Le PDP-10 et ses communautés	86
9.2.1	Un environnement partagé	87
9.2.2	Du terminal au réseau	88
9.2.3	Un héritage durable	89
9.3	Une famille brillante mais isolée	90
10	Le PDP-11	91
10.1	Une architecture plus régulière	92
10.2	L'UNIBUS	94
10.3	Le succès et les limites du PDP-11	95
11	UNIX, C sur PDP-11	99
11.1	Après Multics	100
11.2	Le passage au PDP-11	101
11.3	Un langage pour le système	102
11.4	Expliquer et transmettre	103
12	De la machine au système	105
12.1	Une pluralité de logiciels	106
12.2	La force de DECUS	107
12.3	Le service comme promesse	108
12.4	Changer d'échelle sans perdre son identité	109
IV	L'empire VAX	111
13	Évoluer sans rompre	113
13.1	Étendre l'adresse	114
13.2	Le VAX-11/780	115
13.3	Une famille plutôt qu'un sommet	116
14	Le logiciel comme héritage	117
14.1	Concevoir avec la machine	118

14.2	Migrer sans tout recommencer	119
14.3	La disponibilité comme argument	119
15	Le réseau devient le système	121
15.1	Une entreprise distribuée	122
15.2	Le terminal devient une norme	122
15.3	VAXcluster et le stockage commun	123
15.4	Un monde cohérent, mais de moins en moins fermé	124
16	L'informatique personnelle échappe à DEC	127
16.1	Un marché qui change de règles	127
16.2	Trois réponses au lieu d'une	128
16.3	Le poids de l'organisation	129
16.4	La station de travail	130
16.5	Le logiciel devient déterminant	131
16.6	Quand l'utilisateur choisit l'écosystème	132
16.7	Une force devenue contrainte	133
V	Le dernier pari	135
17	Une culture devenue empire	137
17.1	Des entrepreneurs à l'intérieur de l'entreprise	138
17.2	Le départ d'Harlan Anderson	139
17.3	Ken et la preuve technique	140
17.4	L'utilisateur au centre...	140
18	Alpha contre le temps	143
18.1	Une architecture tournée vers l'avenir	144
18.2	La puissance en démonstration	145
18.3	Une architecture sans répit	145
19	La fin de l'indépendance	147
19.1	Le départ de Ken Olsen	147

19.2 Ce que Compaq veut acheter	148
19.3 Une fin plus complexe qu'un échec	149
Épilogue — Ce que DEC a rendu possible	151
A Chronologie	155
B Repères sur les familles	159
C Glossaire	161
Bibliographie	165
Index général	169

Avant-propos

Digital Equipment Corporation n'est plus une marque familière au grand public. Son nom ne figure ni sur les téléphones, ni sur les ordinateurs portables, ni sur les services en ligne qui occupent aujourd'hui le devant de la scène. Pourtant, une partie décisive de l'informatique moderne s'est construite sur ses machines, dans les laboratoires qui les ont accueillies et parmi les communautés qui ont appris à les détourner de leur destination première.

Écrire l'histoire de **DEC**, ce n'est donc pas dresser le catalogue d'un constructeur disparu. C'est suivre une transformation plus vaste : celle qui conduit de l'ordinateur central, tenu à distance de ses utilisateurs, vers des machines installées au plus près des expériences, des ingénieurs et des programmeurs. Le PDP-1 donne une forme commerciale à l'informatique interactive. Le PDP-8 élargit l'accès au calcul. Le PDP-10 accompagne les communautés du temps partagé. Le PDP-11 devient l'un des terrains sur lesquels grandissent UNIX et le langage C. VAX, VMS et DECnet font ensuite de l'ordinateur un élément d'un système distribué. Alpha montre enfin qu'une entreprise en difficulté peut encore produire une architecture remarquable.

Les machines occupent une place importante dans ce récit, mais elles n'en sont pas les seuls personnages. Il faut aussi rencontrer *Ken Olsen* et *Harlan Anderson* avant qu'ils ne fondent l'entreprise, comprendre ce qu'ils apprennent au **MIT Lincoln Laboratory**, suivre *Gordon Bell*, *Alan Kotok*, *Dave Cutler* et les équipes qui transforment des idées en produits. Il faut surtout rendre leur place aux utilisateurs : étudiants du MIT, chercheurs, techniciens, membres de DECUS, ingénieurs d'usine et programmeurs de **Bell Laboratories**. Ce sont eux qui donnent aux architectures une vie que les seules fiches techniques ne peuvent restituer.

L'ouvrage adopte donc la forme d'une saga industrielle et technique. Les scènes et les anecdotes reposent sur des faits documentés ; aucun dialogue ni détail de décor n'est inventé. La technique est

expliquée lorsqu'elle éclaire une décision, un usage ou une rupture historique. Les notions secondaires sont parfois précisées en note, mais les idées indispensables restent dans le corps du récit.

Les sources visibles ont été volontairement limitées. Une note bibliographique à chaque paragraphe rendrait la lecture pénible sans rendre le texte plus vrai. En revanche, les dates, chiffres, épisodes structurants et affirmations discutables ont été contrôlés à partir de documents primaires et de travaux indépendants. La bibliographie et la notice méthodologique placées en fin d'ouvrage permettent de retrouver les principaux matériaux utilisés.

Le récit s'arrête en 1998, lorsque **Compaq** achève le rachat de **DEC**. Les technologies, les équipes et les systèmes installés poursuivent leur existence au-delà de cette date, mais l'entreprise indépendante a disparu. Ce choix donne au livre une unité : quarante et une années durant lesquelles une petite société installée dans un ancien moulin devient le deuxième constructeur informatique mondial, avant de perdre la maîtrise d'un marché qu'elle avait contribué à transformer.

Le monde d'IBM

En 1957, le mot « ordinateur » n'évoque ni un objet posé sur un bureau, ni un outil que l'on allume pour quelques minutes. Il désigne une installation. Derrière une porte surveillée, des armoires occupent une salle entière, des bandes magnétiques tournent devant des voyants, des lecteurs avalent des paquets de cartes et une imprimante produit les résultats plusieurs heures plus tard. Autour de la machine travaillent opérateurs, programmeurs, techniciens de maintenance et responsables de l'ordonnancement. L'utilisateur ordinaire n'entre pas nécessairement dans la pièce.

Ce monde est dominé par **IBM**. L'entreprise vient des machines à cartes perforées et de l'organisation mécanographique des grandes administrations. Elle maîtrise le commerce, le service, la location des équipements et la relation avec les directions générales. Ses ordinateurs scientifiques de la série 700 et ses machines de gestion donnent une forme industrielle à un secteur encore jeune. Pour un nouveau constructeur, la difficulté n'est donc pas seulement de concevoir une bonne électronique : il faut affronter une organisation mondiale, des équipes de vente, des techniciens présents chez les clients et une réputation déjà installée.

Le contraste entre les deux mondes ne doit toutefois pas être simplifié. **IBM** ne représente pas seulement la lourdeur bureaucratique que les jeunes ingénieurs aiment dénoncer. L'entreprise fournit des machines fiables, une documentation abondante et un service que ses clients peuvent appeler dans plusieurs pays. Elle a compris très tôt qu'un ordinateur n'est pas vendu une seule fois : il faut installer les périphériques, former les opérateurs, remplacer les pièces et garantir la continuité des programmes. Ce savoir industriel constitue une barrière au moins aussi haute que la technique.

Pour les dirigeants d'entreprise, cette organisation est rassurante. Confier la paie, la facturation ou la gestion des stocks à une machine engage en effet l'entreprise pour plusieurs années. Il faut pouvoir compter sur un constructeur capable d'assurer durablement la

maintenance, de fournir de nouveaux périphériques, de faire évoluer les logiciels et d'accompagner l'extension du système.

La domination d'**IBM** ne repose donc pas seulement sur la qualité de ses ordinateurs, mais sur un ensemble cohérent : architecture, location des équipements, cartes perforées, imprimantes, logiciels, réseau commercial et service de maintenance. Un nouvel entrant ne peut espérer rivaliser en proposant simplement de meilleurs circuits. Il doit choisir un autre terrain.

En dehors des entreprises, les universités et les organismes publics dépendent eux aussi de centres de calcul. Les chercheurs réservent du temps, remettent leurs programmes et récupèrent les résultats. Les machines sont parfois remarquablement rapides, mais leur organisation tient l'expérimentation à distance. Cette situation crée une frustration particulière chez ceux qui ont connu les premiers systèmes interactifs du MIT : ils savent qu'une autre relation avec l'ordinateur est techniquement possible, tout en constatant qu'elle reste presque absente du marché.

La machine avant l'utilisateur

Dans les centres de calcul, le traitement par lots s'impose comme une réponse rationnelle à la rareté du matériel. Les programmes et leurs données sont préparés sur cartes ou sur bande, regroupés, puis exécutés les uns après les autres. Cette méthode évite de laisser un équipement coûteux attendre qu'un humain réfléchisse devant sa console. Elle optimise le temps de la machine, mais allonge celui du programmeur. Une erreur de frappe peut n'apparaître qu'après le passage du lot ; il faut alors corriger le paquet, le soumettre de nouveau et attendre une autre exécution.

Le décalage ne choque pas tous les utilisateurs. Pour calculer une paie, trier un fichier de clients ou résoudre une série d'équations préparées à l'avance, le modèle fonctionne. Il devient plus gênant lorsqu'un scientifique veut modifier immédiatement un paramètre, lorsqu'un ingénieur souhaite relier un capteur au calculateur ou lorsqu'un programmeur cherche une erreur en observant pas à pas le comportement de son code. Dans ces situations, le temps humain reprend de la valeur.

La puissance reste également inséparable de ses périphériques. Un processeur isolé ne sert à rien sans lecteurs, imprimantes, bandes et moyens de stockage. Le coût réel d'un système comprend donc des équipements nombreux, des locaux adaptés et une maintenance permanente. Les gros ordinateurs se comptent en centaines de milliers de dollars, parfois en millions. Ils sont achetés ou loués par des organisations capables d'assurer un taux d'utilisation élevé.

Le lot possède sa propre géographie. Le programmeur travaille dans un bureau ou une salle de perforation ; l'opérateur, derrière une autre porte, manipule les supports et surveille la console. Entre eux circule un paquet de cartes accompagné d'instructions. Cette séparation protège la machine contre les erreurs et permet d'enchaîner les travaux, mais elle retire au programmeur la possibilité de regarder le système réagir.

Un cycle de mise au point peut ainsi occuper une journée pour une erreur qui serait corrigée en quelques secondes devant une console. Les programmeurs apprennent à relire soigneusement leur code et à préparer des sorties de diagnostic. Cette rigueur n'est pas inutile, mais elle décourage les essais spontanés. Elle favorise les tâches bien définies, moins les recherches où le problème se précise à mesure que l'on obtient des résultats.

Le traitement par lots n'est donc pas un archaïsme irrationnel. Il est la réponse économique à un ordinateur rare. La rupture viendra lorsqu'un constructeur acceptera de valoriser autrement le temps : non plus seulement le temps de la machine, mais celui de l'ingénieur, du scientifique ou de l'étudiant qui attend devant elle.

Un marché qui s'élargit sans se rapprocher

IBM comprend elle-même que le marché ne peut rester limité aux plus grandes installations. Annoncé en 1959, l'IBM 1401 est proposé en location à partir d'environ 2 500 dollars par mois, bien moins que les grandes configurations scientifiques.

Dans la configuration de base recensée en 1961, comprenant l'unité centrale, le lecteur-perforateur de cartes et l'imprimante, la location mensuelle s'élève précisément à 2 475 dollars. Le même ensemble peut être acheté pour 125 600 dollars. Ce prix augmente

toutefois rapidement lorsque l'installation reçoit davantage de mémoire ou des unités de bandes magnétiques.

Son succès est immense : au milieu des années 1960, la famille 1401 représente, selon **IBM**, plus de la moitié des ordinateurs installés dans le monde. La machine rapproche l'automatisation des entreprises moyennes, mais elle demeure un système de traitement de données organisé autour des cartes perforées, des bandes magnétiques et de l'impression. Elle ne place pas encore le programmeur au centre de l'expérience.¹

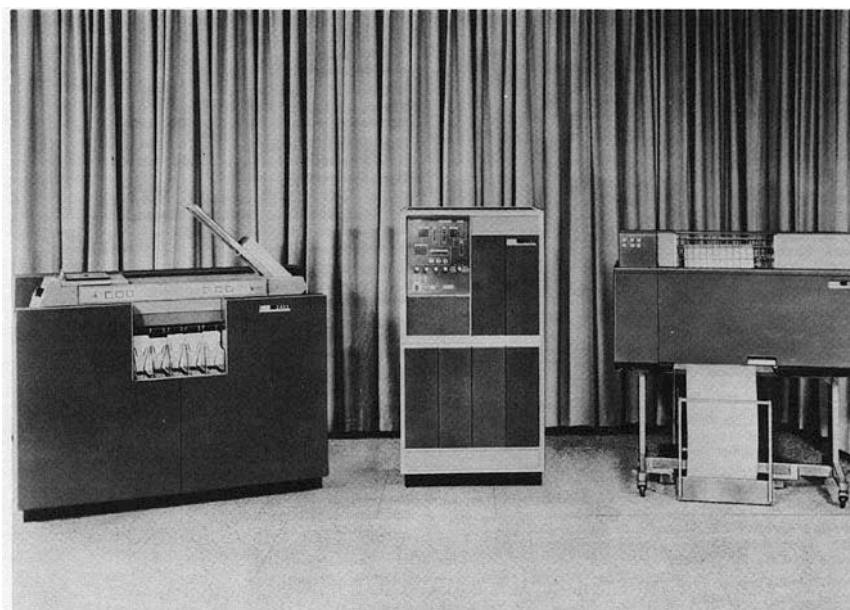


Figure 1 – Configuration de base de l'**IBM** 1401 en 1961 : lecteur-perforateur de cartes 1402, unité centrale 1401 et imprimante 1403.

Source : Ballistic Research Laboratories, 1961. Domaine public, via Wikimedia Commons.

Le succès du 1401 confirme l'existence d'un immense marché en dessous des plus gros calculateurs. Il révèle aussi la force du modèle **IBM** : même une machine plus accessible reste intégrée à une chaîne complète de cartes perforées, de bandes magnétiques, d'imprimantes, de logiciels et de services. Le constructeur ne vend

1. IBM. *The IBM 1401*. IBM Heritage ; **weik-1961-brl** ; Paul E. CERUZZI. *A History of Modern Computing*. 2^e éd. Cambridge, Massachusetts : MIT Press, 2003. ISBN : 978-0-262-53203-7.

pas seulement un ordinateur, mais un environnement dans lequel l'entreprise peut inscrire durablement son activité.

En 1964, le **System/360** pousse cette logique industrielle beaucoup plus loin. **IBM** remplace plusieurs gammes incompatibles par une famille commune, capable de répondre aussi bien aux besoins scientifiques qu'aux applications commerciales. L'investissement est colossal, mais la promesse est simple : un client peut augmenter la puissance de son installation sans abandonner ses programmes, ses données ni une grande partie de ses périphériques.

La compatibilité devient ainsi un avantage stratégique aussi important que la vitesse de calcul. Elle offre aux entreprises une continuité que les constructeurs de machines plus petites ne peuvent désormais ignorer. Quelques années plus tard, ce principe posera à **DEC** une question difficile : comment préserver l'agilité et la simplicité de ses ordinateurs tout en garantissant la continuité attendue d'un grand constructeur ?²

À l'autre extrémité, des ingénieurs commencent à raisonner non en volumes de dossiers traités, mais en secondes de réponse. Un radar, un simulateur ou un appareil de mesure impose son propre rythme. Lorsque l'événement physique change, le calculateur doit le suivre. Le dialogue avec la machine naît moins d'un désir de confort que d'une nécessité opérationnelle. C'est cette nécessité qui donnera à *Ken Olsen* et *Harlan Anderson* une expérience que la plupart des futurs entrepreneurs de l'informatique ne possèdent pas.

Une brèche dans le modèle

Le grand système n'est pourtant pas le seul avenir possible. Les besoins militaires en temps réel, les expériences de radar, les simulateurs et les travaux sur l'interaction graphique imposent une autre relation entre l'homme et la machine. Dans ces environnements, recevoir le résultat le lendemain n'a aucun sens. L'ordinateur doit réagir pendant que l'événement se produit, afficher une situation

2. IBM. *The IBM System/360*. IBM Heritage.

et accepter une commande immédiate.³

C'est dans cette brèche que se forme l'idée de *Ken Olsen*. Au **MIT Lincoln Laboratory**, il rencontre des ordinateurs conçus non pour absorber silencieusement des lots, mais pour être observés, commandés et modifiés par des ingénieurs. À ses côtés, un jeune physicien nommé *Harlan Anderson* découvre le même univers. Ils ne cherchent pas encore à détrôner IBM. Leur ambition est plus précise et, à court terme, plus prudente : construire des éléments électroniques que des techniciens pourront employer directement sur leur établi.

Cette prudence donnera naissance à une entreprise qui, pendant quatre décennies, travaillera à déplacer la frontière entre la salle informatique et l'utilisateur. Son histoire commence avant le premier PDP, avant même que ses fondateurs osent inscrire le mot « ordinateur » dans leur projet.

3. Le *temps réel* ne signifie pas nécessairement que le calcul est très rapide. Il signifie que le système doit répondre avant une échéance imposée par le phénomène qu'il observe ou contrôle.

Première partie

Deux ingénieurs face aux
grands systèmes

Chapitre 1

Deux chemins vers Lincoln

Avant de devenir les fondateurs de **Digital Equipment Corporation**, *Ken Olsen* et *Harlan Anderson* suivent deux parcours qui ne semblent pas devoir se rejoindre. L'un grandit dans le Connecticut, répare des radios et travaille l'été dans un atelier mécanique. L'autre étudie la physique dans l'Illinois et découvre l'informatique à une époque où la discipline se construit encore à partir de schémas, de tubes électroniques et de cours donnés par ceux qui contribuent eux-mêmes à l'inventer.

Leur rencontre ne résulte pas d'un projet entrepreneurial préparé dans une école de commerce. Elle se produit dans un laboratoire chargé de résoudre un problème militaire d'une ampleur considérable. Cette expérience marquera durablement la culture de **DEC** : le goût des systèmes concrets, la confiance accordée aux ingénieurs et la conviction qu'un ordinateur doit pouvoir être utilisé directement comme un instrument de travail, plutôt que rester une machine distante à laquelle on soumet des cartes avant d'attendre les résultats.

1.1 Ken Olsen, le goût du concret

Kenneth H. Olsen naît en 1926 dans le Connecticut. Adolescent, il démonte et répare des postes de radio, activité qui oblige à relier un symptôme visible à un composant caché. Il travaille également dans un atelier mécanique. Ce mélange d'électronique et de fabrication restera au cœur de sa manière de penser : une idée n'acquiert de valeur que lorsqu'elle peut être construite, testée et rendue fiable.

Après un service dans l'US Navy entre 1944 et 1946, il entre au **Massachusetts Institute of Technology**. Il obtient un diplôme de premier cycle en génie électrique en 1950, puis un master en 1952.

Le MIT de l'après-guerre n'est pas seulement une université

prestigieuse, mais aussi l'un des principaux lieux où se rencontrent alors recherche scientifique, défense et industrie. Les travaux menés sur le radar, la mémoire à tores et les calculateurs numériques y entretiennent une continuité entre ces trois domaines.

Ken Olsen se trouve ainsi au milieu d'une génération pour laquelle l'électronique ne se réduit plus à un assemblage de composants isolés : elle devient la base de systèmes capables de recevoir des informations, de les traiter et d'agir en conséquence.¹

Son tempérament est déjà celui d'un constructeur. Il s'intéresse à la fiabilité, aux méthodes de test et aux architectures modulaires. Au lieu d'accepter qu'un ordinateur soit une pièce unique dont chaque panne exige une enquête interminable, il cherche des moyens de séparer les fonctions, d'observer les signaux et de remplacer rapidement ce qui ne fonctionne pas.

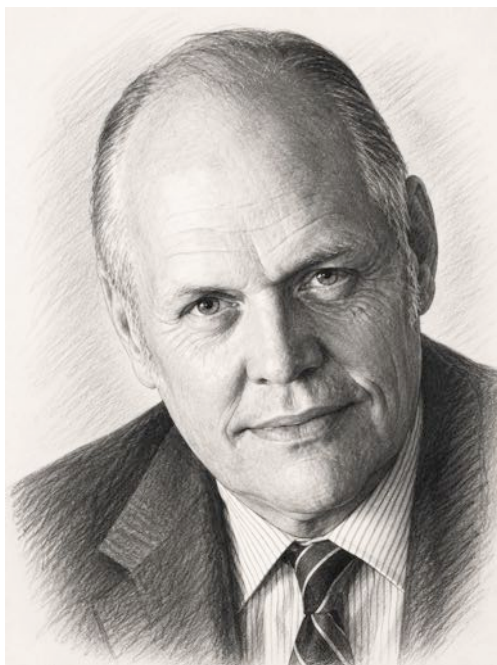


Figure 1.1 – Portrait stylisé de *Ken Olsen*, cofondateur de **DEC**.

Source : Illustration réalisée avec OpenAI. D'après une photographie de *Ken Olsen*.

1. COMPUTER HISTORY MUSEUM. *Ken Olsen* ; MIT NEWS. *Kenneth Olsen, Computing Pioneer and Founder of DEC, Dies at 84*. 2011.

La trajectoire d'Olsen n'est pas celle d'un théoricien qui découvrirait tardivement la fabrication. Il apprend au contraire à juger une conception par ses pannes. Les mémoires à tores, par exemple, promettent une mémoire rapide et non volatile, mais leur mise au point exige des procédés précis et des dispositifs de test capables de détecter des défauts intermittents.

Cette expérience explique aussi son rapport futur aux utilisateurs. Un ingénieur de terrain qui décrit une panne ne doit pas être considéré comme le dernier maillon d'une hiérarchie. Il apporte une information sur le produit. Olsen conservera longtemps l'habitude de regarder les machines, d'interroger ceux qui les réparent et de défendre des décisions à partir d'une preuve matérielle. Cette attention donnera à **DEC** une exigence de qualité, mais elle rendra parfois difficiles les arbitrages qui ne peuvent être tranchés sur un établi.

1.2 Harlan Anderson, de la physique à l'informatique

Harlan E. Anderson naît en 1929 à Freeport, dans l'Illinois. À l'université de l'Illinois, où il étudie la physique, il suit en 1950 des cours liés à l'ILLIAC I, alors en construction.²

L'ordinateur n'est pas encore un produit que l'on apprend à utiliser à partir d'un manuel standardisé. Les étudiants travaillent au contact d'une machine conçue et construite localement. *Harlan Anderson* découvre ainsi l'informatique à la fois par la programmation et par la proximité avec ceux qui réalisent le matériel.

Il obtient un bachelor en 1951 puis un master en physique en 1952. Cette même année, il rejoint le Lincoln Laboratory. Son premier responsable est Ken Olsen. Le lien hiérarchique précède donc de plusieurs années l'association entre les deux hommes. Anderson apporte une formation scientifique et un intérêt marqué pour les

2. L'ILLIAC I est un ordinateur électronique à programme enregistré construit par l'université de l'Illinois selon une architecture dérivée des travaux menés autour de *John von Neumann* à l'Institute for Advanced Study. Équipé de 2 800 tubes à vide et d'une mémoire utilisant des tubes cathodiques, il entre en service en septembre 1952.

usages du calcul ; Olsen possède déjà une expérience plus affirmée de l'électronique et de l'organisation des équipes.³

Leurs caractères ne se confondent pas. Les témoignages disponibles présentent Olsen comme un dirigeant fortement attaché au produit, capable de défendre une intuition technique avec obstination. Anderson joue un rôle important dans la création de l'entreprise, son financement et ses premières relations professionnelles. Cette complémentarité sera précieuse au démarrage, même si leurs trajectoires finiront par diverger.



Figure 1.2 – Portrait stylisé de *Harlan E. Anderson*, cofondateur de **DEC**.

Source : Illustration réalisée avec OpenAI. D'après une photographie de *Harlan E. Anderson*.

Anderson découvre de son côté une discipline encore dépourvue de frontières professionnelles stables. Un physicien peut devenir programmeur, participer au câblage d'une interface puis discuter d'un usage scientifique. Cette mobilité compte : elle forme des ingénieurs qui ne séparent pas nettement le matériel, le logiciel et

3. COMPUTER HISTORY MUSEUM. *Harlan Anderson* ; COMPUTER HISTORY MUSEUM. *Harlan Anderson*.

l'application. Dans la future entreprise, un produit sera souvent conçu à partir du problème d'un client plutôt qu'à partir d'une catégorie établie par le marché.

Il apporte également un tempérament plus extérieur. Là où Olsen aime la profondeur technique et le contrôle du produit, Anderson se montre attentif aux réseaux professionnels, au financement et à la manière dont une jeune société peut se présenter. Le contraste ne doit pas être transformé en opposition romanesque : les deux hommes sont d'abord des ingénieurs et partagent la même confiance dans les petites équipes. Mais leurs priorités ne sont pas identiques, et cette différence réapparaîtra lorsque **DEC** cessera d'être une aventure de quelques dizaines de personnes.

1.3 Un rendez-vous dans l'histoire

En 1952, aucun des deux hommes ne peut prévoir qu'ils fonderont cinq ans plus tard l'un des grands constructeurs informatiques. Leur horizon immédiat est le Lincoln Laboratory, créé par le MIT pour répondre à la menace des bombardiers soviétiques. Le laboratoire rassemble des mathématiciens, des électroniciens, des spécialistes du radar et des programmeurs autour du projet SAGE.

Ce contexte compte davantage que la simple ligne biographique. Les futurs fondateurs de **DEC** apprennent dans un environnement où l'ordinateur doit communiquer avec des radars, afficher des informations et recevoir des ordres en quelques secondes. Ils voient aussi combien les systèmes deviennent difficiles à construire et à maintenir lorsqu'ils sont conçus comme des ensembles monolithiques.

Leur rencontre possède enfin une dimension générationnelle. Ils arrivent à l'informatique au moment où aucune carrière standard n'existe encore. Les écoles ne forment pas des architectes de processeurs au sens moderne ; les langages et les systèmes d'exploitation ne constituent pas encore des industries séparées. Ceux qui construisent les machines doivent souvent en définir simultanément les circuits, les méthodes de programmation et les usages.

Au Lincoln Laboratory, *Ken Olsen* et *Harlan Anderson* vont découvrir une informatique soumise à des exigences nouvelles. Il ne suffit plus d'effectuer correctement un calcul : les machines

doivent traiter des informations en temps utile, communiquer avec d'autres équipements et rester opérationnelles malgré la complexité croissante de leurs circuits.

Cette expérience fournira aux deux hommes bien davantage qu'une ligne dans leur parcours professionnel. Elle formera leur conception de l'ordinateur, leur manière d'organiser le travail technique et, quelques années plus tard, les fondations de leur propre entreprise.

Chapitre 2

Le laboratoire des possibles

Le **MIT Lincoln Laboratory** naît en 1951 dans l'urgence de la guerre froide. Les États-Unis cherchent alors à détecter suffisamment tôt les bombardiers susceptibles d'approcher leur territoire par le nord et à coordonner une réponse à l'échelle du continent. Le futur système SAGE doit relier des radars, des centres de calcul, des écrans et des réseaux de communication. Aucun ordinateur existant ne peut, à lui seul, remplir une telle mission. Une grande partie de l'infrastructure doit être inventée en même temps que le système.

Pour *Ken Olsen* et *Harlan Anderson*, le laboratoire constitue une école incomparable. Ils y voient l'informatique sortir des limites du calcul numérique traditionnel. Un écran ne sert plus seulement à présenter un résultat : il représente une situation en constante évolution. Un opérateur ne se contente plus d'attendre une impression : il sélectionne une information et reçoit une réponse. Le logiciel atteint une complexité qui impose de nouvelles méthodes d'organisation. Quant à la fiabilité, elle cesse d'être une qualité simplement souhaitable pour devenir une condition de fonctionnement.

2.1 Du calcul à l'interaction

Le projet *Whirlwind*, lancé au MIT avant la création du Lincoln Laboratory, avait déjà rompu avec l'image de l'ordinateur destiné principalement au calcul de tables. Conçu à l'origine pour un simulateur de vol, il devait réagir immédiatement aux actions d'un pilote. Cette exigence conduisit à privilégier un fonctionnement rapide et régulier, une mémoire adaptée au traitement en temps réel ainsi que des dispositifs permettant d'afficher et de manipuler les informations.

Le système SAGE (*Semi-Automatic Ground Environment*),

conçu pour la surveillance et la défense aériennes des États-Unis, reprend cette orientation et l'étend à une échelle nationale. Des données provenant de nombreux radars doivent être transmises aux centres de contrôle, traitées par des ordinateurs, puis présentées aux opérateurs sous une forme directement exploitable. Son organisation et son fonctionnement seront examinés plus loin.

Le TX-0, construit au milieu des années 1950, pousse plus loin l'emploi des transistors et de la mémoire à tores. Les transistors remplissent ici le rôle d'interrupteurs électroniques capables de représenter et de manipuler les états binaires. Plus petits, moins fragiles et moins gourmands en énergie que les tubes à vide, ils permettent d'envisager des ordinateurs plus compacts, plus fiables et moins difficiles à refroidir.

La mémoire repose sur de minuscules anneaux de ferrite, appelés *tors*, traversés par des fils électriques. Le sens d'aimantation de chaque tore permet de représenter un bit à 0 ou à 1. Ces éléments conservent leur état même lorsque l'alimentation est interrompue et offrent une robustesse bien adaptée aux exigences des nouveaux calculateurs.

Le TX-0 sert notamment à éprouver les circuits transistorisés ainsi qu'une partie des technologies de mémoire destinées au TX-2. Son écran et son crayon lumineux permettent d'agir directement sur les informations affichées. L'ordinateur n'est plus seulement un équipement auquel on confie un calcul soigneusement préparé : il devient une machine que l'on peut observer et manipuler en temps réel.

Le Lincoln Laboratory présente aujourd'hui le TX-0 comme l'un des premiers ordinateurs entièrement transistorisés et comme une étape majeure de l'informatique interactive.¹

Au Lincoln Laboratory, *Ken Olsen* développe notamment le *Memory Test Computer*, destiné à éprouver les dispositifs de mémoire, et participe à la conception matérielle des TX-0 et TX-2. *Harlan Anderson*, recruté par Olsen, travaille également sur le projet TX-0.

Les deux hommes découvrent ainsi une idée dont l'industrie ne

1. MIT LINCOLN LABORATORY. *Lincoln Laboratory History*. Massachusetts Institute of Technology ; MIT LINCOLN LABORATORY. *Interactive Supercomputing : From Whirlwind to TX-2*. Massachusetts Institute of Technology.

mesure pas encore toute la portée : lorsqu'un utilisateur obtient rapidement une réponse, il modifie sa manière de travailler. Il essaie, observe le résultat, corrige son programme et recommence. Le programme devient moins un objet confié à la machine que le moyen d'entretenir avec elle un échange continu.

Dans les salles de contrôle de SAGE, l'écran et le dispositif de pointage annoncent un avenir qui n'est pas encore celui de l'ordinateur personnel. L'opérateur observe des symboles représentant la situation aérienne, sélectionne une trace et transmet une décision. La machine ne se contente plus de recevoir une pile de cartes : elle conserve une représentation du monde extérieur et la met continuellement à jour à partir des données qui lui parviennent.

Cette relation transforme la notion même de programme. Le logiciel n'est plus seulement une suite de calculs qui s'achève par la production d'un résultat. Il devient un système qui doit rester actif, recevoir des événements, modifier son état et continuer à répondre sans interruption.

Pour les ingénieurs, ce fonctionnement permanent impose une nouvelle discipline. Une panne interrompant un calcul scientifique peut entraîner la perte de plusieurs heures de travail. Dans un système de défense aérienne, elle peut priver temporairement les opérateurs d'une partie des informations nécessaires à leur mission. Les techniques de test, de redondance, de surveillance et de diagnostic acquièrent donc une importance essentielle.

Ken Olsen apprend ainsi que la disponibilité d'une machine ne dépend pas seulement de la qualité de ses composants. Elle doit être prévue dès la conception de son architecture, notamment dans la manière de découper le système en sous-ensembles, d'observer son fonctionnement et de localiser rapidement une défaillance.

2.2 La leçon de la modularité

SAGE est aussi une leçon par sa démesure. Une installation composée de milliers de composants ne peut être entretenue uniquement par quelques spécialistes capables d'en comprendre chaque détail. Il faut découper le système en fonctions testables, normaliser les connexions et prévoir le remplacement des éléments défaillants. La

modularité n'est donc pas un simple choix esthétique : elle réduit le temps de diagnostic et permet à plusieurs équipes de travailler parallèlement.

Olsen développe au Lincoln Laboratory des équipements de test et s'intéresse à l'emploi de blocs électroniques réutilisables. Cette démarche prépare directement les premiers produits de **DEC**. Un ingénieur qui souhaite construire un compteur, un additionneur ou un contrôleur ne devrait pas avoir à reprendre chaque fois la totalité du schéma. Il devrait pouvoir associer des fonctions déjà connues et éprouvées, comme il combine différents instruments sur un établi.

Le laboratoire forme également ses ingénieurs à une pratique fondée sur le prototype. Une idée doit pouvoir être câblée, mesurée et confrontée à l'usage. Sa réalisation matérielle permet de révéler des difficultés qu'un schéma ou une note technique ne suffisent pas toujours à prévoir. Cette expérience contribue à former chez Olsen une méthode que l'on retrouvera plus tard chez **DEC** : construire, tester, observer les défaillances, puis corriger la conception à partir des résultats obtenus.

L'histoire ultérieure du TX-0 révèle également les possibilités contenues dans cette nouvelle forme d'informatique. Transférée sur le campus du MIT en juillet 1958, alors qu'Olsen et Anderson ont déjà fondé **DEC**, la machine devient accessible à des chercheurs et à des étudiants.

Ceux-ci peuvent approcher sa console, observer immédiatement ses réactions et modifier leurs programmes au fil des essais. Ils ne la considèrent plus seulement comme un équipement coûteux auquel il faut remettre un travail soigneusement préparé. Ils en font un instrument d'exploration, utilisé directement et parfois avec une liberté inhabituelle.

Le laboratoire fournit enfin un réseau humain. Les ingénieurs passent d'un projet à un autre, connaissent les fournisseurs et apprennent à reconnaître ceux qui savent transformer un schéma en matériel fiable. Les futurs fondateurs ne quitteront donc pas le MIT avec une idée isolée. Ils emporteront des méthodes, des relations professionnelles et la conviction que la région de Boston peut soutenir une nouvelle industrie électronique.

2.3 Quitter une institution puissante

Le Lincoln Laboratory offre des moyens considérables, mais sa mission militaire et la taille croissante de ses projets limitent l'autonomie individuelle. À mesure que les systèmes deviennent plus vastes, le délai entre une idée et sa réalisation peut s'allonger. Olsen souhaite fabriquer des équipements plus petits, destinés aux ingénieurs, aux laboratoires et aux installations industrielles. Anderson partage la conviction qu'un marché existe pour de tels produits et, à terme, pour des systèmes moins coûteux que les grands ordinateurs.

La décision de partir mûrit dans un contexte où le mot « ordinateur » inquiète les investisseurs. Les machines alors connues exigent des millions de dollars, des équipes nombreuses et plusieurs années de développement. Olsen et Anderson ne peuvent donc pas présenter leur projet comme la construction immédiate d'un concurrent aux grands systèmes commercialisés par les entreprises déjà établies. Ils doivent réduire l'ambition visible pour préserver leur ambition réelle : rapprocher l'informatique de ceux qui l'utilisent.

Partir reste pourtant risqué. Les constructeurs existants disposent de capitaux, de brevets, d'usines et d'équipes commerciales. Les deux hommes ne peuvent réunir immédiatement les sommes nécessaires à la conception d'un ordinateur complet. Leur stratégie consiste donc à commencer par ce qu'ils savent construire, ce que les laboratoires peuvent déjà acheter et ce que leur expérience rend crédible : des modules logiques.

Un module logique est un sous-ensemble électronique réalisant une fonction déterminée. Certains effectuent une opération logique élémentaire ; d'autres contiennent une bascule capable de conserver un bit, un compteur ou un générateur d'impulsions. Ces fonctions ne sont pas encore réunies dans des circuits intégrés, mais obtenues à partir de transistors, de diodes, de résistances et de condensateurs.

En associant plusieurs modules normalisés, un ingénieur peut construire un instrument de mesure, un dispositif de commande ou une partie d'ordinateur sans devoir concevoir chaque fonction depuis le début. Il peut également remplacer plus facilement un élément défaillant, ce qui prolonge directement les méthodes de

test et de maintenance apprises au Lincoln Laboratory.

Cette solution correspond aux besoins des laboratoires. Ceux-ci ne cherchent pas nécessairement à acquérir un ordinateur complet, mais peuvent acheter quelques fonctions électroniques fiables et les intégrer à leurs propres installations. Pour Olsen et Anderson, les modules constituent donc un premier produit réaliste, fondé sur leurs compétences et susceptible de trouver rapidement des clients.

Cette prudence ne traduit pas une absence d'ambition. Elle permet au contraire de limiter l'investissement initial tout en préparant une entreprise capable de construire, à terme, ses propres ordinateurs. Il reste cependant à convaincre des investisseurs et à transformer cette idée en société.

Chapitre 3

La naissance de DEC

Au printemps 1957, *Ken Olsen* et *Harlan E. Anderson* ne disposent ni d'une usine, ni d'un produit terminé, ni d'une fortune personnelle capable de financer plusieurs années de développement. Ils possèdent en revanche une expérience rare des circuits transistorisés, acquise au **MIT Lincoln Laboratory**, ainsi qu'une idée assez précise du type de machines qu'ils souhaitent un jour construire.

Leur ambition se heurte toutefois à une difficulté immédiate. Concevoir un ordinateur complet exige des moyens considérables. Au matériel proprement dit doivent s'ajouter les périphériques, les logiciels, la documentation, la fabrication en série et le service après-vente. Les deux ingénieurs doivent donc présenter aux investisseurs un projet susceptible de produire rapidement des revenus, sans renoncer pour autant à leurs objectifs à plus long terme.

Le 27 mai 1957, ils soumettent leur proposition à l'**American Research and Development Corporation**, généralement désignée par le sigle ARD. Quelques mois plus tard, en août, **Digital Equipment Corporation** commence ses activités à Maynard, dans le Massachusetts.

L'entreprise ne naît donc pas dans un garage devenu légendaire, mais dans le cadre encore nouveau du capital-risque américain.¹

3.1 Un projet prudent

L'interlocuteur principal des deux ingénieurs est le général *Georges Doriot*, professeur à Harvard et dirigeant d'ARD. Fondée en 1946, cette société appartient aux premières expériences

1. Le *capital-risque* consiste à financer une jeune entreprise en échange d'une part de son capital. L'investisseur accepte un risque élevé d'échec dans l'espoir qu'une réussite prendra une valeur considérable.

américaines de capital-risque organisé. Elle rassemble l'argent de plusieurs institutions afin de financer des entreprises dont la valeur repose moins sur des installations existantes que sur une technologie, une équipe et une possibilité de croissance.

La proposition remise à ARD ne promet pas de concurrencer immédiatement **IBM**. *Ken Olsen* et *Harlan E. Anderson* prévoient d'abord de vendre des modules électroniques aux laboratoires, aux fabricants d'équipements et aux ingénieurs chargés de construire des systèmes de mesure, de contrôle ou de traitement numérique.

Ces modules constituent des circuits logiques prêts à l'emploi. Ils permettent à leurs utilisateurs de réaliser des équipements spécialisés sans devoir concevoir eux-mêmes chaque élément électronique. Le marché paraît assez limité pour ne pas provoquer une réaction immédiate des grands constructeurs, mais suffisamment étendu pour assurer les premiers revenus d'une petite entreprise.

Cette stratégie présente surtout l'avantage de reporter la construction d'un ordinateur complet. Un tel projet nécessiterait non seulement une unité centrale, mais également des périphériques, des logiciels, une documentation abondante, une organisation commerciale et un service de maintenance. Les modules peuvent au contraire être conçus, fabriqués et vendus séparément, à mesure que les commandes arrivent.

ARD investit 70 000 dollars et reçoit 70 % du capital de la nouvelle société. Le montant reste modeste à l'échelle de l'industrie informatique, mais il permet aux fondateurs de louer des locaux, d'acheter leurs premiers équipements et de recruter une petite équipe.²

Le financement ne leur accorde cependant pas une liberté totale. Le conseil d'administration conserve un droit de regard important sur les décisions de la jeune entreprise. *Georges Doriot* attend un projet organisé, un produit rapidement commercialisable et une gestion suffisamment rigoureuse pour limiter les risques.

Le nom retenu traduit lui-même cette prudence. **Digital Equipment Corporation** contient le mot anglais *equipment*, et non le

2. COMPUTER HISTORY MUSEUM, *Ken Olsen*, cf. note 1 ; COMPUTER HISTORY MUSEUM, *Harlan Anderson*, cf. note 3 ; COMPUTER HISTORY MUSEUM. *Digital Equipment Corporation Founded*. 1957.

mot *computer*.³

La société se présente ainsi comme un fournisseur d'équipements numériques, plutôt que comme un nouveau constructeur d'ordinateurs. Cette formulation évite de placer immédiatement la jeune entreprise face à **IBM**, **Sperry Rand** ou **RCA**. Elle laisse pourtant ouverte la possibilité de réaliser plus tard des systèmes complets.

Le plan de développement comprend en réalité deux étapes. La première repose sur la fabrication et la vente des modules. La seconde prévoit d'utiliser les compétences, les circuits et les revenus ainsi obtenus pour concevoir une machine interactive. Pour y parvenir, **DEC** doit d'abord démontrer sa capacité à fabriquer des produits fiables, à respecter ses délais et à trouver des clients.

Le partage initial du capital rappelle la fragilité de la position des fondateurs. Leur réussite dépendra moins du contrôle financier qu'ils détiennent en 1957 que de la croissance future de l'entreprise. Chaque commande livrée et chaque série de modules vendue contribuent ainsi à renforcer une autonomie encore limitée.

3.2 L'ancienne filature

En août 1957, **DEC** s'installe dans une partie de l'ancienne filature de Maynard, un vaste complexe industriel construit à partir du milieu du XIX^e siècle au bord de la rivière Assabet. Le site, autrefois consacré à la production textile, demeure couramment désigné sous le nom de *The Mill*.

Les premiers locaux loués par l'entreprise couvrent 8 680 pieds carrés, soit environ 806 mètres carrés. Cette surface reste modeste, mais elle suffit pour réunir les bureaux, un atelier et les premiers équipements de production. Le réaménagement de bâtiments existants coûte également bien moins cher que la construction d'une usine neuve.

3. Le mot français « ordinateur », dans son sens informatique actuel, est proposé en 1955 par le philologue *Jacques Perret*, à la demande d'**IBM France**, qui cherche un équivalent au terme anglais *computer*. Cette création terminologique française n'a donc joué aucun rôle dans le choix du nom de **DEC**, effectué aux États-Unis. Le terme « ordinateur » est néanmoins employé dans cet ouvrage comme équivalent français courant.



Figure 3.1 – L’ancien complexe industriel de Maynard dans lequel **DEC** installe son siège en 1957. Sa tour d’horloge devient progressivement l’un des symboles de l’entreprise.

Source : Illustration réalisée avec OpenAI. D’après une photographie historique du site industriel de Maynard.

Le contraste est saisissant : une entreprise issue des laboratoires militaires et tournée vers les techniques électroniques les plus récentes prend place derrière les murs d’une ancienne usine textile. Les briques, les poutres et la tour d’horloge appartiennent au paysage industriel du siècle précédent, tandis que les ateliers commencent à produire des circuits transistorisés.⁴

Maynard se trouve à une certaine distance de Cambridge, mais reste dans l’environnement industriel de la région de Boston. Autour de la Route 128 se concentrent alors des laboratoires, des fournisseurs et un nombre croissant d’entreprises spécialisées dans l’électronique. **DEC** peut ainsi bénéficier de cette proximité technique

4. COMPUTER HISTORY MUSEUM. *Digital Equipment Corporation* ; DIGITAL EQUIPMENT CORPORATION, INFORMATION RESEARCH SERVICES. *DIGITAL Computing History Timeline, 1957–1997*. 1998.

sans supporter le coût d'une implantation au centre de Cambridge.

L'ancien complexe offre également une possibilité d'agrandissement. À mesure que l'entreprise se développe, ses bureaux, ses laboratoires et ses ateliers occupent une part croissante des bâtiments disponibles. Des espaces peuvent être réaménagés et de nouvelles équipes accueillies sans imposer le transfert immédiat de l'ensemble des activités vers un autre site.

Cette implantation rassemble sur un même lieu une grande partie des opérations de la société. La conception des circuits, la réalisation des prototypes, l'assemblage et les essais se trouvent géographiquement rapprochés. Durant les premières années, cette proximité facilite les échanges entre les ingénieurs et les personnes chargées de fabriquer et de tester les équipements.

Elle contribue aussi à donner à **DEC** une culture fortement technique. Les produits sont encore assez peu nombreux pour que leurs concepteurs suivent directement leur passage du schéma au prototype, puis à la fabrication. L'entreprise conserve ainsi quelque chose de l'atelier d'ingénieurs dont elle est issue, même lorsque ses activités commencent à prendre une dimension industrielle.

Le site ne reste donc pas un simple décor. Au fil des années, l'ancienne filature accompagne matériellement la croissance de l'entreprise. Son réseau de bâtiments, de couloirs, d'escaliers et de passerelles forme progressivement un vaste campus industriel.

Maynard devient ainsi bien davantage que l'adresse administrative de **DEC**. L'entreprise y conserve son siège pendant plusieurs décennies et marque profondément la vie locale. La tour d'horloge qui domine les anciens bâtiments de briques finit par devenir l'un des symboles les plus reconnaissables de la société.

3.3 Les premiers pas

La direction de la nouvelle entreprise repose d'abord sur une répartition relativement simple des responsabilités. *Ken Olsen* prend en charge l'orientation technique, la conception des produits et leur industrialisation. Il accorde une attention particulière à la fiabilité, aux coûts et à la possibilité de fabriquer les circuits en série. *Harlan E. Anderson*, nommé vice-président exécutif, intervient

davantage dans la gestion, les finances, les relations extérieures et le développement commercial.

Autour d'eux, la petite équipe doit assurer presque toute la chaîne de production. Il faut concevoir les circuits, acheter les composants, préparer les cartes, les assembler, les tester, rédiger leur documentation et rechercher des clients. Les fonctions qu'une grande entreprise répartit entre plusieurs services demeurent encore étroitement associées.

Cette organisation correspond également aux attentes des premiers acheteurs. Les modules ne sont pas destinés au grand public, mais à des ingénieurs et à des techniciens capables de lire un schéma, de mesurer un signal et d'intégrer les cartes dans leurs propres installations. La vente prend donc la forme d'un échange technique autant que commercial.

La documentation doit indiquer précisément les caractéristiques électriques, les connexions et les fonctions de chaque module. Une carte doit pouvoir être intégrée dans un équipement plus vaste sans obliger le client à en redécouvrir le fonctionnement. La qualité de cette documentation devient ainsi presque aussi importante que celle du circuit lui-même.

La fiabilité constitue un autre enjeu essentiel. Un module défaillant peut immobiliser un appareil de mesure, perturber une expérience ou interrompre un système de contrôle. Pour convaincre les laboratoires et les industriels, **DEC** doit fournir des produits reproductibles, correctement testés et susceptibles d'être remplacés rapidement.

Les premières ventes permettent à l'entreprise de démontrer qu'elle sait transformer une expérience acquise au laboratoire en produits réellement commercialisables. Cette étape exige des compétences différentes de celles nécessaires à la réalisation d'un prototype unique. Les circuits doivent être fabriqués en série, rester conformes à leurs spécifications et être accompagnés d'une documentation utilisable par des clients extérieurs.

En juillet 1958, à la fin de son premier exercice fiscal, **DEC** a vendu pour environ 94 000 dollars d'équipements numériques et emploie déjà une soixantaine de personnes. La société reste minuscule face aux grands constructeurs, mais ces résultats valident

son premier modèle économique. Elle a trouvé des clients et montré qu'elle était capable de concevoir, de fabriquer, de documenter et de vendre ses propres produits.

Le succès reste pourtant fragile. Le conseil d'administration hésite encore à engager les sommes nécessaires au développement d'un ordinateur complet. La jeune société doit continuer à prouver que son activité peut devenir rentable et durable avant de franchir une nouvelle étape.

Les modules ne constituent cependant pas une simple solution d'attente. Leur conception oblige **DEC** à maîtriser les fonctions logiques, la normalisation des signaux, les connexions entre cartes, l'alimentation, les essais et la fabrication en série. Chacune de ces compétences sera nécessaire lorsqu'il faudra réunir plusieurs centaines de circuits au sein d'une même machine.

Les premiers produits de **DEC** sont donc des briques électroniques, mais ils préparent déjà la suite. En apprenant à les concevoir, à les produire et à les associer, la jeune entreprise acquiert progressivement les moyens de construire des systèmes beaucoup plus ambitieux.

Chapitre 4

Commencer par les modules

Les premiers produits de **DEC** ne ressemblent pas à des ordinateurs miniatures. Ce sont des boîtiers et des cartes qui accomplissent une fonction logique précise : amplifier un signal, compter des impulsions, mémoriser un état ou réaliser une opération booléenne. Un chercheur peut les poser sur un établi, les relier par des cordons et construire rapidement un dispositif adapté à son expérience.

Cette apparente modestie donne à la jeune entreprise trois avantages. Les modules correspondent à un besoin réel, ils peuvent être vendus sans développer un environnement logiciel et leur fabrication enseigne à **DEC** la discipline industrielle qui manque souvent aux prototypes de laboratoire. Chaque carte doit fonctionner comme la précédente, résister au transport et être comprise à partir d'une documentation claire.

4.1 Des briques pour les ingénieurs

Les *Laboratory Modules* sont conçus pour l'expérimentation et le contrôle. Leur présentation rend les connexions et la fonction logique directement visibles. L'utilisateur peut modifier le câblage, observer le résultat puis réemployer les mêmes éléments dans un autre projet.

La figure 4.1 montre un exemple de ces premiers produits : un module de type *flip-flop* 201, commercialisé vers 1960. Une bascule est un circuit capable de conserver un état binaire, donc un bit valant 0 ou 1. Le circuit électronique est placé dans un boîtier métallique, tandis que les bornes disposées sur la face avant permettent de le relier à d'autres modules au moyen de fils cavaliers. En associant ainsi des bascules, des portes logiques, des compteurs ou des générateurs d'impulsions, un ingénieur peut construire progressivement un dispositif de commande, un instrument scientifique

ou une partie d'ordinateur.

Ces modules figurent parmi les tout premiers produits commercialisés par **DEC**. Ils prolongent directement la culture du Lincoln Laboratory : décomposer une fonction complexe en blocs identifiables, testables et remplaçables. Ils sont bientôt suivis par les *System Building Blocks*, qui remplissent des fonctions électriques comparables, mais utilisent un système de connecteurs standardisé mieux adapté à l'intégration permanente dans des ensembles plus importants, notamment des ordinateurs.¹

Les *System Modules*, commercialisés après les *Laboratory Modules*, sont destinés à la construction d'ensembles plus permanents. Ils ne se relient plus librement par des cordons placés sur leur face avant, mais s'enfichent dans un châssis équipé d'un fond de panier. Celui-ci distribue l'alimentation et établit les connexions entre les différents modules. Une carte peut ainsi être retirée, remplacée ou associée à d'autres sans qu'il soit nécessaire de reconstruire tout le système.

Cette organisation convient à des équipements appelés à fonctionner durablement : contrôleurs, testeurs de mémoire, instruments scientifiques, automatismes industriels ou systèmes logiques plus complexes. **DEC** présente ces modules comme des « Digital Building Blocks », des briques numériques que l'ingénieur peut combiner pour construire progressivement une machine adaptée à ses besoins.

La figure 4.2 montre un *System Module* de type *Inverter* 1103-H. Un inverseur réalise l'une des fonctions les plus simples de la logique numérique : il produit l'état opposé à celui qu'il reçoit. Une entrée correspondant à 1 donne donc une sortie à 0, tandis qu'une entrée à 0 produit une sortie à 1.

Le module réunit six inverseurs indépendants, ce qui explique la répétition de plusieurs groupes de composants presque identiques. Il ne constitue pas pour autant un registre de six bits : chacune de ses six voies traite simplement un signal distinct. Elles peuvent éventuellement être utilisées en parallèle pour agir sur six lignes binaires, mais elles peuvent tout aussi bien appartenir à des parties différentes d'un même système.

1. Le module conservé par le Computer History Museum est daté approximativement de 1960 et porte le numéro d'inventaire 102633142.

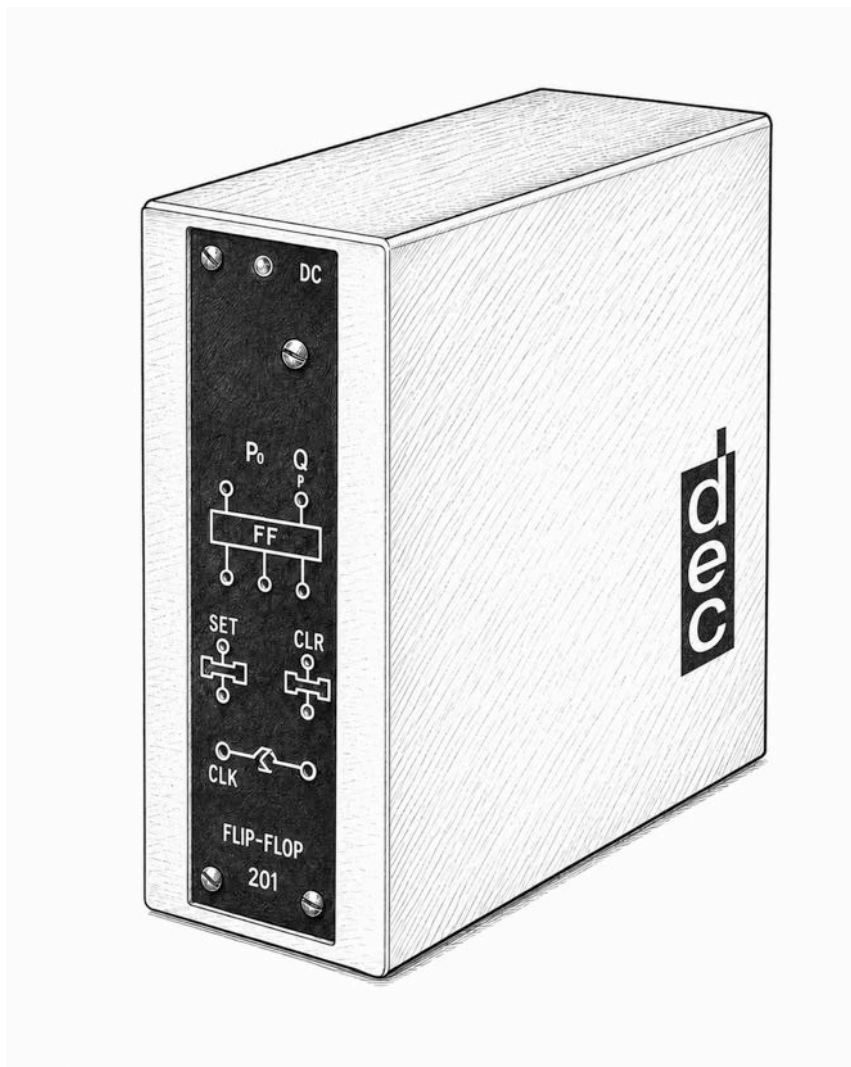


Figure 4.1 – Représentation illustrée d'un module logique de laboratoire DEC de type *flip-flop* 201, vers 1960.

Source : Illustration originale réalisée à partir de documentation historique. Module de référence : Computer History Museum, accession 102633142.

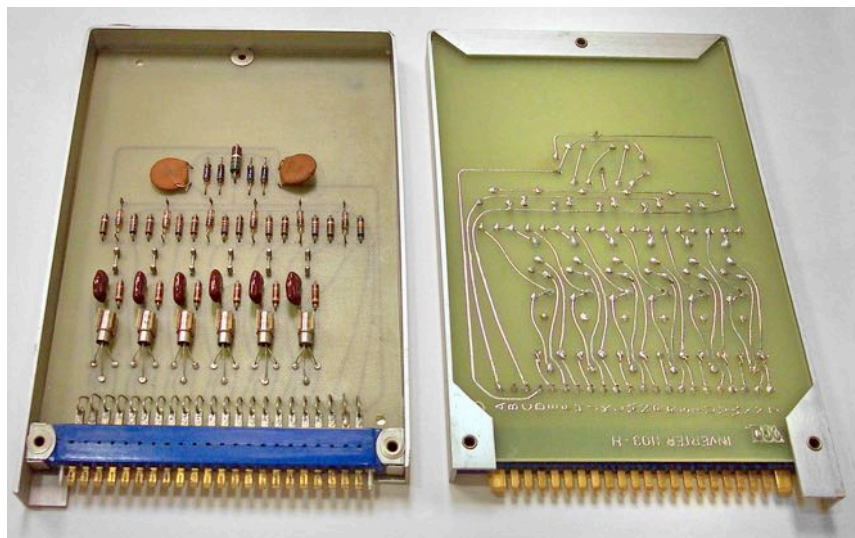


Figure 4.2 – Module système DEC de type *Inverter* 1103-H, présenté côté composants et côté câblage.

Source : RTC, via Wikimedia Commons. Licence CC BY-SA 3.0.

À cette époque, les fonctions logiques ne sont pas encore couramment disponibles sous la forme de circuits intégrés compacts contenant des portes déjà constituées. Pour réaliser un inverseur, il faut assembler des composants élémentaires : transistors, diodes, résistances et condensateurs. La répétition visible sur la photographie correspond donc à la reproduction matérielle de la même fonction logique six fois sur le même module.

La face arrière rend également perceptible une méthode de fabrication encore largement manuelle. Les connexions ne sont pas formées par des pistes de cuivre dessinées sur un circuit imprimé moderne, mais par des fils métalliques découpés, courbés et soudés individuellement entre les composants et le connecteur. Leur tracé épouse la position réelle des broches, contourne les obstacles et cherche à maintenir des liaisons suffisamment courtes.

Ces courbes et ces légères irrégularités donnent au câblage une apparence presque organique. Elles laissent voir le geste de la personne qui a assemblé le module, là où les cartes électroniques actuelles présentent des pistes régulières calculées par logiciel, gravées industriellement et souvent assemblées automatiquement. Le module se situe ainsi à une période intermédiaire : sa fonction est

standardisée et reproductible, mais sa réalisation conserve encore une part visible de travail artisanal.

Cette construction modulaire facilite également la maintenance. Lorsqu'une fonction présente une défaillance, le technicien peut identifier la carte concernée, la retirer du châssis et la remplacer sans immobiliser longtemps l'ensemble de l'installation. La modularité n'est donc pas seulement une méthode de conception : elle devient un moyen d'améliorer la disponibilité et la réparabilité des systèmes.²

4.2 Une documentation qui enseigne

Pour présenter ses produits, **DEC** publie dès le début des années 1960 des catalogues et des bulletins techniques consacrés à ses différentes familles de modules. Certaines publications proposent une présentation abrégée de la gamme, tandis que d'autres détaillent les caractéristiques électriques et les possibilités d'association de chaque circuit.

Ces documents ne se limitent pas à montrer un produit et à en indiquer la référence. Ils précisent la fonction logique des modules, leurs niveaux électriques, leur alimentation, leurs connexions et leurs temps de propagation. Ils fournissent également des schémas et des exemples montrant comment plusieurs modules peuvent être associés pour construire un compteur, un registre, un circuit de commande ou une interface.

Pour un ingénieur, cette documentation tient donc à la fois du catalogue commercial et du manuel de conception numérique. **DEC** ne vend pas une boîte fermée dont le fonctionnement resterait caché : l'entreprise fournit des briques clairement décrites et les connaissances nécessaires pour les intégrer à un système conçu par le client.

2. DIGITAL EQUIPMENT CORPORATION, INFORMATION RESEARCH SERVICES, cf. note 4.

4.3 Apprendre la série

La production de modules oblige **DEC** à résoudre des problèmes moins spectaculaires que l'architecture d'un ordinateur, mais tout aussi déterminants : choix des composants, régularité des approvisionnements, méthodes de soudure, contrôle qualité et organisation des tests. Une différence minime entre deux séries peut provoquer chez un client une panne difficile à reproduire. L'entreprise met donc au point des bancs de test et des procédures destinés à vérifier chaque fonction avant la livraison.

L'avantage de **DEC** ne réside pas d'abord dans l'invention de composants entièrement nouveaux. Les transistors, les diodes, les résistances et les principes logiques employés sont déjà connus. La valeur du produit vient davantage de la manière dont l'entreprise les sélectionne, les assemble, les caractérise et garantit leur fonctionnement. Un client n'achète pas seulement quelques composants réunis sur une carte : il achète une fonction logique documentée, testée et reproductible.

Cette maîtrise industrielle distingue progressivement **DEC** de petits ateliers capables de produire un prototype, mais moins bien préparés à livrer plusieurs exemplaires identiques et à les maintenir dans le temps. La documentation précise les niveaux électriques, les connexions et les conditions d'utilisation. Lorsqu'un problème apparaît, les ingénieurs de Maynard peuvent en rechercher la cause, proposer une modification ou remplacer le module concerné. La confiance du client repose donc autant sur le suivi technique que sur le circuit lui-même.

L'apprentissage de la série transforme également le travail des concepteurs. Dans un laboratoire, un montage peut être ajusté jusqu'à ce qu'un exemplaire fonctionne. Dans une production industrielle, la conception doit supporter les variations normales des composants et pouvoir être testée par une personne qui n'a pas participé à sa mise au point. Les points de mesure, les limites électriques, les procédures de contrôle et les instructions de maintenance deviennent ainsi des parties intégrantes du produit.

Cette expérience prépare directement les futurs ordinateurs. Un PDP ne sera pas un prototype entièrement distinct de ce qui le

précède, mais un assemblage organisé de fonctions dont les caractéristiques sont déjà connues. Le catalogue et les stocks de l'entreprise forment progressivement une véritable bibliothèque matérielle. Selon une remarque attribuée avec humour à *Ben Gurley*, le PDP-1 aurait été conçu en partie pour utiliser ce qui se trouvait déjà dans les réserves de Maynard. L'anecdote résume une contrainte réelle : l'élégance d'une architecture doit toujours composer avec le prix, les délais et les possibilités de fabrication.³

La robustesse des premiers PDP ne repose donc pas sur une accumulation d'astuces impossibles à reproduire. Le processeur est nouveau par l'organisation de ses fonctions, mais une grande partie de sa logique s'appuie sur des modules déjà fabriqués, mesurés et éprouvés. La modularité réduit ainsi le risque technique et financier du passage à l'ordinateur complet.

Les modules permettent enfin à **DEC** de construire un marché avant même que celui des petits ordinateurs soit clairement identifié. Ils entrent dans des laboratoires, des universités et des entreprises qui découvrent la qualité de fabrication, la documentation et la disponibilité des équipes de Maynard. Un client satisfait d'un compteur, d'un contrôleur ou d'un ensemble logique connaît déjà le constructeur lorsqu'il envisage ensuite l'achat d'un système complet.

Le succès initial de **DEC** ne repose donc pas sur une rupture technologique unique. Il résulte d'un ensemble plus difficile à imiter : une conception compréhensible, une fabrication régulière, des produits testés, une documentation détaillée et un suivi assuré par des ingénieurs capables de dialoguer directement avec les utilisateurs. C'est ce savoir-faire, autant que l'électronique elle-même, qui donne à la jeune entreprise sa place sur le marché.

4.4 La tentation de la machine complète

À mesure que les commandes progressent, la frontière entre module et ordinateur devient moins nette. Certaines installations réunissent déjà assez de mémoire, de logique de commande et d'interfaces pour former de petites machines spécialisées. Des clients

3. COMPUTER HISTORY MUSEUM. *Origins of the PDP-1*.

demandent des ensembles plus intégrés, tandis que les ingénieurs de **DEC** savent, grâce à leur expérience du TX-0, qu'une architecture interactive peut être construite à partir des technologies qu'ils maîtrisent désormais.

Le passage à un système complet reste toutefois risqué. Concevoir un ordinateur ne consiste pas seulement à assembler davantage de circuits. Il faut fournir des périphériques, écrire des logiciels, former les utilisateurs et assurer la maintenance pendant plusieurs années. Une machine vendue trop tôt peut mobiliser durablement les ingénieurs et absorber les ressources de la jeune société.

Le conseil d'administration craint surtout que **DEC** ne s'engage sur le terrain des grands constructeurs sans disposer de leurs capitaux ni de leur organisation commerciale. *Ken Olsen* et *Harlan E. Anderson* doivent donc montrer qu'il ne s'agit pas d'imiter **IBM** en plus petit, mais de répondre à des besoins que les grandes installations couvrent mal : calcul scientifique à proximité de l'utilisateur, contrôle d'expériences, traitement en temps réel et interaction directe avec la machine.

Le vocabulaire fournit une partie de la solution. Le futur système ne sera pas présenté comme un *computer*, terme associé à un investissement lourd et à une infrastructure spécialisée, mais comme un *Programmed Data Processor*. Cette expression le rapproche des équipements scientifiques et des systèmes de contrôle que les laboratoires achètent déjà. Elle évite d'affirmer trop ouvertement que **DEC** entre sur le marché des ordinateurs, même si l'acronyme PDP deviendra bientôt l'un des plus connus de l'histoire informatique.

Le débat porte ainsi sur l'identité même de l'entreprise. Les modules ont permis à **DEC** d'acquérir une technologie, un savoir-faire industriel et un réseau de clients. Rester uniquement fournisseur de composants reviendrait désormais à abandonner à d'autres la possibilité de transformer ces acquis en systèmes complets. Construire une machine signifie prendre davantage de risques, mais aussi donner une cohérence aux briques logiques que la société fabrique depuis sa création.

En 1959, **DEC** possède les circuits, l'expérience de production et les premiers utilisateurs susceptibles d'adopter une telle machine. Il lui manque encore un ingénieur capable d'organiser ces ressources

autour d'une architecture cohérente. Le recrutement de *Ben Gurley*, qui a travaillé sur des systèmes informatiques et des circuits à grande vitesse, transforme alors une ambition prudente en véritable projet industriel.

Deuxième partie

Rapprocher la machine

Chapitre 5

Le PDP-1, vendre une idée

Le premier ordinateur de **DEC** ne naît pas d'une étude de marché démontrant l'existence d'une nouvelle catégorie de produits. Il repose sur une conviction : une machine inspirée des systèmes expérimentaux du Lincoln Laboratory peut intéresser des laboratoires qui n'ont ni les moyens ni l'envie de dépendre d'un grand centre de calcul.



Figure 5.1 – PDP-1 présenté dans le laboratoire de démonstration du Computer History Museum à Mountain View. On distingue l'unité centrale, le pupitre de commande, le télétype et l'écran à tube cathodique.

Source : Alexey Komarov, travail personnel, via Wikimedia Commons. Licence CC BY-SA 4.0.

Cette intuition doit néanmoins être transformée en un produit industriel. La machine doit être suffisamment fiable pour être vendue, assez simple pour être fabriquée par une jeune entreprise et assez différente pour ne pas se retrouver dans une comparaison directe avec les grands ordinateurs d'**IBM**.

Ben Gurley joue ici un rôle décisif. Ingénieur rapide et inventif, il conçoit l'architecture du PDP-1 en quelques mois à partir des techniques et des modules déjà maîtrisés par **DEC**. La machine utilise des mots de 18 bits, une mémoire à tores de 4096 mots dans sa configuration de base et un jeu d'instructions volontairement compact.

La largeur de 18 bits rattache le PDP-1 à une tradition de calcul scientifique et aux travaux menés autour du TX-0. Chaque mot peut contenir une instruction ou une valeur numérique. Avec 4096 mots, la mémoire paraît minuscule selon les critères actuels, mais elle représente déjà plus de 73 000 bits directement accessibles au processeur. Elle suffit pour des programmes scientifiques et interactifs, à condition d'écrire avec économie.

Le PDP-1 n'est pas conçu pour battre les plus gros calculateurs sur tous les terrains. Il doit fournir une puissance suffisante dans un volume et à un prix accessibles à une université, à un laboratoire ou à une équipe de recherche. Son architecture privilégie notamment les échanges avec les périphériques et la possibilité de réagir à des événements extérieurs.

Un dispositif expérimental peut ainsi signaler au processeur qu'une donnée est disponible ou qu'une opération vient de se terminer. Grâce au mécanisme d'interruption, le programme suspend momentanément son travail, traite l'événement, puis reprend son exécution. Cette capacité compte davantage pour les utilisateurs visés qu'une longue liste d'opérations arithmétiques.

Le panneau avant donne une forme visible à cette philosophie. Ses rangées de voyants indiquent l'état des registres et de certaines parties de la machine. Les commutateurs permettent d'introduire une instruction, de placer une valeur en mémoire, d'examiner son contenu ou de lancer l'exécution.

Un ingénieur peut donc démarrer et diagnostiquer le PDP-1 sans attendre qu'un système d'exploitation en prenne le contrôle. Cette utilisation exige une connaissance précise de la machine, mais elle établit aussi une relation de confiance : l'ordinateur n'est pas une boîte interdite dont seuls les techniciens du constructeur comprendraient le fonctionnement.

Ben Gurley dessine ainsi une machine suffisamment simple pour

être comprise par une petite équipe et suffisamment ouverte pour que chaque client l'adapte à ses besoins. Là où une grande architecture tente de prévoir toutes les applications, le PDP-1 fournit un noyau autour duquel l'utilisateur peut construire son propre environnement.

5.1 La preuve par la démonstration

Le prototype est présenté à la fin de 1959 lors de l'Eastern Joint Computer Conference de Boston. Dans un salon rempli de machines, de chiffres et d'annonces commerciales, le PDP-1 se distingue surtout par son comportement. Un utilisateur peut se placer devant la console, lancer un programme, observer l'activité des registres et recevoir une réponse immédiate.

L'affichage à tube cathodique Type 30 renforce cette impression. Il ne présente pas encore des pages de texte ou des images composées de millions de pixels. Il trace des points et des segments lumineux commandés directement par l'ordinateur. Associé à un programme adapté, il permet de visualiser un résultat, de suivre l'évolution d'un calcul ou d'interagir avec une représentation graphique.

Cette présence transforme la manière de vendre la machine. Les grands systèmes sont souvent décrits par leur vitesse de calcul, leur débit de lecture ou le volume de données qu'ils peuvent traiter. Le PDP-1 peut être montré comme un instrument de laboratoire : on exécute un programme, on modifie un paramètre, puis on observe immédiatement le résultat.

Pour une société qui ne possède pas encore une force commerciale comparable à celle des constructeurs établis, cette démonstration est essentielle. Le visiteur doit comprendre la différence en quelques minutes. Une figure qui se déplace sur l'écran ou un calcul relancé depuis la console rendent l'argument concret : le prospect ne voit pas seulement ce que la machine calcule, mais la manière dont il pourrait travailler avec elle.

La démonstration reste toutefois une épreuve technique. Le programme doit fonctionner, l'affichage doit être correctement réglé et la console doit supporter les manipulations du public. Chaque présentation expose les faiblesses éventuelles du prototype, mais

elle donne aussi à **DEC** une crédibilité que les slogans ne suffiraient pas à produire.

Le prix de base, d'environ 120 000 dollars, demeure considérable. Le PDP-1 n'annonce pas encore l'ordinateur individuel. Son coût reste toutefois très inférieur à celui des grandes installations scientifiques, qui peuvent atteindre plusieurs millions de dollars et nécessiter des locaux spécialement aménagés.

Un département universitaire ou un laboratoire peut ainsi envisager de posséder sa propre machine, au lieu d'acheter du temps de calcul auprès d'un centre partagé. Ce déplacement est fondamental : l'ordinateur se rapproche de l'équipe qui l'utilise et peut rester disponible pour ses expériences.

5.2 BBN, le premier client

Le premier PDP-1 de production est vendu à **Bolt Beranek and Newman**, société de recherche et de conseil installée à Cambridge et connue notamment pour ses travaux en acoustique.

Ed Fredkin, qui a vu le prototype, recommande son acquisition afin de soutenir les recherches de *J. C. R. Licklider* sur la psychoacoustique et les relations entre l'être humain et la machine. Le système est livré en 1960.

BBN n'achète pas un produit fermé. Ses ingénieurs écrivent les outils qui leur manquent, adaptent le système à leurs recherches et entretiennent des relations directes avec les équipes de Maynard. *Ed Fredkin* réalise notamment FRAP, un assembleur destiné au PDP-1.

Un assembleur traduit des instructions symboliques écrites par le programmeur en codes compréhensibles par le processeur. Au lieu de manipuler directement des suites numériques, l'utilisateur peut employer des noms d'opérations, des étiquettes et des adresses symboliques. FRAP rend ainsi la programmation du PDP-1 plus lisible et plus rapide.¹

Cette coopération préfigure une relation qui deviendra carac-

1. COMPUTER HISTORY MUSEUM. *Ed Fredkin* ; COMPUTER HISTORY MUSEUM. *Scientific Applications on the PDP-1*.

téristique de **DEC**. Le client technique n'est pas seulement un utilisateur qu'il faut former. Il peut devenir une source d'idées, produire des logiciels utiles à d'autres installations et parfois contribuer indirectement à l'évolution de la gamme.

L'ouverture des entrées-sorties joue ici un rôle fondamental. Un laboratoire peut connecter des convertisseurs, des instruments de mesure, des commandes ou des dispositifs expérimentaux sans demander au constructeur de redessiner l'ensemble de l'ordinateur. La diversité des usages du PDP-1 dépend donc moins de sa seule puissance de calcul que de sa capacité à accueillir des équipements nouveaux.

Chez BBN, l'ordinateur entre dans un milieu où les frontières entre recherche fondamentale, contrats publics et applications industrielles restent mobiles. Les travaux de *J. C. R. Licklider* sur la communication entre l'être humain et la machine trouvent dans le PDP-1 un instrument plus souple qu'un centre de calcul traditionnel.

Cette relation oblige également Maynard à écouter ses utilisateurs. Une amélioration conçue pour un client peut devenir une option commerciale. Un programme peut circuler vers une autre installation. Un technicien de terrain peut rapporter un usage auquel les concepteurs n'avaient pas pensé.

Le PDP-1 installe ainsi un cycle que **DEC** reproduira longtemps : fournir une plate-forme compréhensible et ouverte, laisser les utilisateurs l'étendre, puis intégrer une partie de leurs idées dans les logiciels, les options ou les machines suivantes.

Les faibles volumes de vente masquent donc une réussite stratégique. Chaque installation fonctionne comme un laboratoire de marché. Les usages graphiques, le temps partagé, la musique, la simulation et le contrôle d'expériences ne constituent pas encore des secteurs clairement séparés. Le PDP-1 permet à **DEC** de les observer avant qu'ils ne deviennent des catégories commerciales établies.

5.3 Une autre manière de compter

Le PDP-1 n'est produit qu'à quelques dizaines d'exemplaires. Mesuré selon les critères de la production de masse, ce résultat paraît modeste. Pour **DEC**, son importance est pourtant considérable : il prouve que la société sait concevoir un ordinateur complet, le fabriquer, l'installer et accompagner ses utilisateurs.

La machine donne également une identité au nom PDP. Les modèles suivants ne seront pas de simples versions plus rapides du premier. **DEC** explorera plusieurs largeurs de mots, plusieurs technologies et plusieurs marchés. Cette diversité provoquera plus tard des problèmes de compatibilité, mais elle exprime d'abord une grande liberté de conception : chaque architecture peut être adaptée à un besoin particulier.

Le succès le plus durable du PDP-1 ne se mesure cependant ni chez BBN ni dans les chiffres de vente. Il commence lorsque **DEC** décide de ne pas vendre son deuxième exemplaire. La société le confie au département de génie électrique du MIT, là même où ses fondateurs ont appris à considérer l'ordinateur comme un instrument interactif.

Le geste paraît généreux. Il est aussi stratégiquement remarquable. Placé entre les mains d'étudiants et de chercheurs capables de modifier ses programmes, le PDP-1 va devenir le centre d'une culture nouvelle. Autour de lui se développeront des outils, des expériences graphiques, des programmes musicaux et un jeu qui marquera durablement l'histoire de l'informatique.

Chapitre 6

Le PDP-1 au MIT

En 1961, une jeune entreprise qui n'a encore vendu qu'un seul exemplaire de son ordinateur décide d'offrir le suivant. Le bénéficiaire n'est pas choisi au hasard. Le PDP-1 rejoint le Research Laboratory for Electronics du MIT, institution située au croisement de la recherche, de l'enseignement et des premiers travaux sur l'informatique interactive.

Le don du deuxième PDP-1 de production est présenté par les archives du Computer History Museum comme le début d'une politique de soutien aux établissements qui forment les futurs professionnels de l'informatique. **DEC** ne place donc pas seulement une machine dans une université prestigieuse. Elle la confie à des étudiants et à des chercheurs susceptibles de développer des logiciels, d'inventer de nouveaux usages et d'emporter ensuite cette expérience dans les laboratoires et les entreprises où ils travailleront.¹

Le geste possède aussi une valeur symbolique. **DEC** revient vers l'institution dont sont issus ses fondateurs, mais elle n'y retourne plus comme groupe de chercheurs ou fournisseur de circuits. Elle apporte un ordinateur commercial conçu à Maynard et destiné à vivre en dehors de l'entreprise. Pour une société fondée quatre ans plus tôt, voir sa première machine installée au MIT constitue une reconnaissance technique particulièrement visible.

Le calcul économique ne peut pas être établi comme celui d'une campagne publicitaire moderne. Personne ne sait combien de ventes futures seront directement provoquées par le don. L'intuition est néanmoins solide : un étudiant formé sur une machine acquiert des habitudes, conserve des programmes et développe une préférence pour une certaine manière de travailler. Lorsqu'il rejoint ensuite un

1. COMPUTER HISTORY MUSEUM. *Invitation to a Presentation of a PDP Computer to Massachusetts Institute of Technology*. 1961 ; COMPUTER HISTORY MUSEUM. *Invitation to the Presentation of a PDP-1 to MIT*. 1961.

laboratoire ou une entreprise, il connaît déjà les produits de **DEC** et sait expliquer ce qu'ils permettent.

Une machine interactive ne se vend d'ailleurs pas seulement par sa vitesse ou par la capacité de sa mémoire. Elle se vend aussi par les pratiques qu'elle rend possibles. Offrir le matériel à une communauté capable d'inventer ces pratiques revient à financer indirectement la formation, la démonstration et le développement logiciel.

6.1 Pourquoi offrir une machine rare ?

À première vue, le choix paraît coûteux. Un PDP-1 représente des composants, des semaines de fabrication et une vente possible. **DEC** ne dispose pourtant ni des budgets publicitaires ni de la force commerciale des grands constructeurs. Une machine réellement utilisée dans une université peut accomplir davantage qu'une brochure : elle forme des programmeurs, produit des démonstrations et donne au constructeur une légitimité scientifique.

La décision correspond également à l'expérience de *Ken Olsen* et de *Harlan E. Anderson*. Ils ont eux-mêmes travaillé dans un environnement où l'on pouvait approcher le matériel, observer les signaux et comprendre le fonctionnement d'un système. Ils savent qu'une machine inaccessible, réservée à quelques opérateurs, ne produit pas la même relation qu'un ordinateur placé à proximité des chercheurs.

Jack Dennis, professeur et ancien membre du Tech Model Railroad Club, joue ici un rôle essentiel. Après avoir encadré l'accès au TX-0, il accepte que des étudiants compétents utilisent le PDP-1 en dehors d'un programme de recherche entièrement défini à l'avance. Le temps machine reste précieux et les projets officiels demeurent prioritaires, mais l'initiative individuelle reçoit une place inhabituelle.

L'accès n'est donc pas libre au sens absolu. Le PDP-1 appartient à un laboratoire, les utilisateurs doivent prouver qu'ils comprennent la machine et chacun doit respecter le travail des autres. Cette contrainte donne tout son relief au rôle de *Jack Dennis* : il ne

distribue pas simplement des heures de calcul, il reconnaît une compétence et accepte qu'un travail sans objectif contractuel immédiat puisse produire quelque chose d'utile.

Cette confiance ne produit pas une organisation parfaitement ordonnée. L'accès à la machine demeure privilégié, les projets officiels restent prioritaires et les hackers entretiennent avec les règles institutionnelles une relation parfois distante. Ils acceptent les contraintes nécessaires au fonctionnement du laboratoire, mais supportent mal les procédures qui leur paraissent empêcher de comprendre ou d'améliorer le système.

Le partage des programmes constitue en revanche une pratique essentielle. Les logiciels sont conservés sur des rubans perforés, rangés notamment dans un tiroir à proximité de la machine. Un utilisateur peut y prendre un programme, l'exécuter, en étudier le fonctionnement, lui emprunter quelques instructions ou lui ajouter une nouvelle fonction. La version améliorée rejoint ensuite les outils disponibles pour les autres.

Le programme n'est donc pas considéré comme une propriété qu'il faudrait soustraire au regard des pairs. Sa valeur augmente lorsqu'il circule, lorsqu'il peut être compris et lorsqu'un autre programmeur parvient à l'améliorer. Cette pratique concrète incarne l'un des principes que *Steven Levy* identifiera plus tard dans l'éthique hacker : l'information doit rester accessible à ceux qui peuvent en faire quelque chose.²

6.2 Une machine presque nue

Lorsque le PDP-1 arrive au MIT, il ne dispose pas encore de l'environnement logiciel qu'un lecteur contemporain associe à un ordinateur. Il faut un assembleur, des programmes de débogage, des routines d'entrée-sortie et des outils capables d'exploiter l'affichage. Chez un grand constructeur, ces éléments auraient été attendus du fabricant. Au MIT, leur absence devient une invitation.

Les utilisateurs développent les programmes dont ils ont besoin.

2. Steven LEVY. *Hackers : Heroes of the Computer Revolution*. Garden City, New York : Anchor Press/Doubleday, 1984.

Les outils circulent, sont corrigés, améliorés et parfois transmis à **DEC** ou à d'autres installations. La jeune entreprise reçoit ainsi des logiciels, des retours d'expérience et des idées qu'une équipe interne encore réduite n'aurait pas pu produire avec la même diversité.

Cette pauvreté initiale possède un effet paradoxal. Elle impose beaucoup de travail, mais rend chaque contribution immédiatement visible. Les logiciels ne proviennent pas d'un service anonyme : ils restent associés aux personnes qui les conçoivent et les partagent.

Autour d'*Alan Kotok*, *Peter Samson*, *Bob Saunders* et *Bob Wagner*, une petite équipe adapte ainsi au PDP-1 un assembleur utilisable par toute la communauté. *Jack Dennis* et *Alan Kotok* sont associés aux premiers outils de débogage interactif, tandis que *Peter Samson* explore la musique, la mise en forme du texte et l'affichage graphique.

Un assembleur, un débogueur ou une routine d'affichage modifie immédiatement le quotidien de tous les utilisateurs. Un programme écrit pendant la nuit, perforé sur un ruban puis déposé dans le tiroir commun peut devenir l'outil du laboratoire dès le lendemain. Chacun peut alors l'utiliser, en examiner le fonctionnement et tenter de l'améliorer.³

Le PDP-1 transforme aussi l'apprentissage de l'architecture. Les étudiants peuvent observer les voyants du panneau, introduire une instruction, modifier un mot en mémoire puis mesurer directement l'effet de leur intervention. Les registres, les adresses et les interruptions cessent d'être des notions abstraites décrites dans un manuel : elles deviennent des états visibles et manipulables.

Cette relation attire une communauté déjà préparée à comprendre les systèmes de l'intérieur.

6.3 La rencontre avec les hackers

Au MIT, le mot *hacker* ne désigne pas encore un intrus qui pénètre frauduleusement dans un système informatique. Il qualifie un étudiant capable de comprendre une construction complexe, d'en découvrir les possibilités cachées et de produire une solution

3. COMPUTER HISTORY MUSEUM. *Alan Kotok*; LEVY, cf. note 2.

aussi ingénieuse qu'inattendue.

Le Tech Model Railroad Club, notamment son comité *Signals and Power*, constitue l'un des lieux où se forme cet état d'esprit. Ses membres travaillent sur les relais, les circuits de commande et les équipements téléphoniques récupérés qui contrôlent leur réseau ferroviaire miniature. Ils apprennent à suivre un signal, à isoler une panne et à modifier une architecture en observant son comportement réel.

Le passage du local du TMRC à la console du TX-0, puis à celle du PDP-1, est donc moins brutal qu'il n'y paraît. Dans les deux cas, un système complexe doit être compris à partir de ses éléments et de leurs interactions. Un montage élégant se reconnaît à l'économie de ses détours, à sa clarté et à l'ingéniosité de la solution.

Le mot *hack* porte alors une idée de performance inventive. Un bon *hack* résout un problème, mais le fait avec une simplicité, une audace ou une surprise qui mérite d'être racontée. Sa valeur vient autant de la compréhension acquise que de l'admiration des pairs.

Cette morale informelle favorise la circulation des outils. Garder pour soi un programme utile priverait la communauté de la possibilité de l'étudier et de l'améliorer. Le partage ne relève pas encore d'une licence formelle ou d'une théorie du logiciel libre ; il constitue une pratique rendue nécessaire par la rareté de la machine et par l'intérêt commun des utilisateurs.

Le PDP-1 convient particulièrement à cette culture. Ses voyants permettent de repérer une boucle qui ne se termine pas, son écran révèle immédiatement une erreur graphique et ses rubans perforés peuvent passer d'une main à l'autre. La machine reste extrêmement coûteuse, mais son usage possède quelque chose de personnel : pendant le temps qui lui est accordé, le programmeur dispose réellement du processeur et peut observer directement ses réactions.

6.4 Un week-end pour fabriquer un outil

Alan Kotok, étudiant en génie électrique et membre du TMRC, devient rapidement l'une des figures de cette communauté. Le PDP-1 dispose encore de peu de logiciels. *Jack Dennis* envisage d'utiliser FRAP, l'assembleur écrit chez BBN par *Ed Fredkin*. *Alan*

Kotok propose plutôt d'adapter au PDP-1 un outil inspiré de celui que les étudiants connaissent déjà sur le TX-0.

Le défi prend la forme d'un pari. Six programmeurs se réunissent pendant un week-end et consacrent environ 250 heures de travail cumulées à l'écriture du nouvel assembleur. Ils se relaient devant la console, écrivent des routines, vérifient les résultats et corrigent les erreurs. Le lundi, le programme fonctionne.⁴

L'assembleur est capable d'assembler son propre code source : une fois une première version opérationnelle obtenue, celle-ci peut traduire le programme symbolique qui la décrit et produire une nouvelle version exécutable d'elle-même. Cette propriété constitue une étape importante, car l'outil n'est plus seulement construit à l'aide de méthodes provisoires : il peut désormais servir à son propre développement.

L'épisode montre la différence entre une équipe constituée par une direction et une communauté mobilisée autour d'un besoin immédiat. Aucun service n'a défini à l'avance les rôles de chacun. Certains écrivent des routines, d'autres les relisent ou les testent, tandis que les suivants attendent leur tour devant la machine. La pression vient moins d'une hiérarchie que du défi et de la volonté de mettre rapidement un outil commun à la disposition de tous.

Cette performance ne doit pas être entièrement idéalisée. Elle repose sur l'accès privilégié à une machine extrêmement coûteuse et sur la disponibilité d'étudiants capables de lui consacrer une grande partie de leur week-end. Elle révèle néanmoins une forme de production collective dans laquelle le programme circule, est examiné par plusieurs personnes et s'améliore par corrections successives.

L'assembleur réduit considérablement la distance entre l'idée du programmeur et son exécution. Au lieu d'écrire directement des suites de nombres représentant les instructions du processeur, l'utilisateur emploie des noms d'opérations, des étiquettes et des adresses symboliques. Le programme devient plus lisible, plus facile à modifier et plus simple à partager.

L'assemblage ne supprime cependant pas les erreurs. Pour les rechercher, les utilisateurs disposent notamment de DDT, pour *DEC*

4. COMPUTER HISTORY MUSEUM, *Alan Kotok*, cf. note 3 ; LEVY, cf. note 2.

Debugging Tape. Chargé à partir d'un ruban perforé et commandé depuis le télétype, ce programme permet d'examiner et de modifier le contenu de certaines cases mémoire. Il peut également rechercher une configuration de bits, afficher ou perforer une zone mémoire et comparer le contenu d'un ruban avec celui qui se trouve dans la machine.

DDT peut en outre lire la table des symboles produite par l'assembleur. Le programmeur ne doit donc plus nécessairement rechercher une erreur uniquement à partir d'adresses numériques en notation octale : il peut retrouver les noms et les repères employés dans son programme. L'assembleur et le débogueur forment ainsi un environnement cohérent : le premier traduit le programme, tandis que le second permet d'en observer et d'en corriger le fonctionnement.

Le débogage devient alors une activité directement liée à l'état de la machine. Le programmeur peut arrêter son programme, examiner une instruction ou une donnée en mémoire, modifier une valeur, puis relancer l'essai. Une correction satisfaisante peut ensuite être enregistrée sur un nouveau ruban perforé et remise à la disposition des autres utilisateurs.

L'erreur n'est plus seulement découverte après l'impression d'un long listing ou après un nouveau passage différé dans un centre de calcul. Elle peut être étudiée devant l'ordinateur, au moment où le programme s'exécute. Le logiciel devient un objet dont on observe les états et que l'on transforme progressivement. Cette manière interactive de programmer et de déboguer dépassera largement le cadre du PDP-1.

6.5 *Spacewar !*

À l'automne 1961, *Steve Russell*, *Martin Graetz* et *Wayne Wiatanen* imaginent un programme qui exploiterait véritablement l'affichage du PDP-1. Un simple dessin démonstratif ne leur suffit pas. Ils veulent une action continue, des règles compréhensibles et un spectacle que l'on ait envie de regarder. Le projet devient *Spacewar !*.

Deux vaisseaux s'affrontent autour d'une étoile dont la gravité modifie leur trajectoire. Les joueurs commandent la rotation, la poussée et le tir. Le programme est rapidement amélioré par plusieurs membres de la communauté. *Peter Samson* ajoute un ciel étoilé fondé sur des données astronomiques, tandis que *Dan Edwards* affine le calcul de la gravitation.

Alan Kotok et *Bob Saunders* construisent des boîtiers de commande afin que les joueurs ne soient plus obligés d'actionner frénétiquement les commutateurs de la console. Le logiciel révèle ici une limite matérielle et provoque la création d'une nouvelle interface.

Le jeu ne constitue donc pas une parenthèse étrangère à la recherche. Il oblige ses auteurs à résoudre des problèmes de temps de réponse, de calcul de trajectoire, d'affichage et de commandes simultanées. Il maintient la machine sous une charge continue et rend immédiatement visibles les défauts de l'interface.

Sa démonstration possède aussi une puissance sociale particulière. Autour de l'écran, les spectateurs comprennent immédiatement ce qui se produit. Ils n'ont pas besoin de lire un tableau de performances : ils voient deux personnes agir, la machine répondre et un monde simulé conserver ses règles.

La force historique de *Spacewar!* tient autant à cette évidence qu'à son statut de jeu précoce. Le programme circule vers d'autres installations PDP-1, où il est copié, modifié et utilisé comme démonstration. **DEC** comprend qu'un jeu peut révéler la vitesse, l'affichage et les entrées-sorties de sa machine mieux qu'une longue présentation commerciale.⁵

L'épisode apporte surtout une leçon durable : les usages importants d'une technologie ne sont pas toujours ceux que son constructeur avait prévus. En ouvrant les interfaces et en laissant les utilisateurs expérimenter, **DEC** leur donne la possibilité d'inventer une culture autour de ses produits.

5. COMPUTER HISTORY MUSEUM. *Spacewar!*

6.6 Partager sans perdre l'interaction

Le succès de cette utilisation personnelle à l'échelle d'un laboratoire fait rapidement apparaître une limite : plusieurs personnes veulent accéder à la même machine. Le temps partagé cherche à donner à chacune l'impression qu'elle dispose de l'ordinateur, alors que le processeur passe rapidement d'un travail à l'autre.⁶

Chez BBN, *Ed Fredkin* et *Jack Dennis* développent un système utilisant une mémoire à tambour pour conserver et échanger les programmes des différents utilisateurs. Une démonstration a lieu en septembre 1962 sur le premier PDP-1 livré à BBN. Des expériences apparentées sont conduites au MIT et à Stanford.⁷

Le modèle du centre de calcul est ainsi renversé. Il ne s'agit plus de rassembler des travaux non interactifs afin de les exécuter successivement, mais de partager une machine tout en préservant le dialogue avec chacun de ses utilisateurs.

Cette évolution transforme aussi l'ordinateur en espace commun. Les programmes et les fichiers doivent être protégés, les ressources réparties et les erreurs d'un utilisateur empêchées de détruire le travail des autres. Une question de performance devient alors une question d'organisation collective.

Les futures machines 36 bits de **DEC** reprendront cette ambition à une échelle beaucoup plus grande. Les utilisateurs qui ont appris à attendre leur tour devant le PDP-1 voudront bientôt rester connectés depuis un terminal. L'interaction individuelle avec la machine se transformera progressivement en conversation collective.

6.7 Ce que le MIT rend à DEC

Le don de 1961 revient vers Maynard sous plusieurs formes. Il fournit des logiciels, des démonstrations, des idées d'interface et un vivier de recrutement. *Alan Kotok* rejoint **DEC** en 1962 et participera ensuite au développement des machines 36 bits. D'autres

6. Le *temps partagé* répartit le temps du processeur entre plusieurs sessions interactives. Lorsqu'un utilisateur attend une saisie ou une opération lente, le système peut exécuter le programme d'un autre.

7. COMPUTER HISTORY MUSEUM. *Timesharing on the PDP-1*.

étudiants, chercheurs et visiteurs emportent avec eux une familiarité avec les produits de la société.

La stratégie dépasse ainsi le seul cas du MIT. Lorsqu'un diplômé rejoint un laboratoire ou une entreprise, il connaît déjà les conventions, les outils et la manière de travailler de **DEC**. Le constructeur ne vend pas seulement une machine : il contribue à former le marché qui saura ensuite l'utiliser.

Pour la jeune société, l'expérience compense également une faiblesse structurelle. **DEC** ne peut pas encore employer assez de programmeurs pour couvrir tous les usages possibles. En plaçant ses machines entre les mains de communautés compétentes, elle obtient une bibliothèque de logiciels et d'expériences plus diverse que celle qu'aurait conçue une équipe centralisée.

Cette méthode ne garantit ni l'homogénéité des programmes ni la simplicité du support. Elle installe cependant une conviction qui marquera longtemps l'entreprise : la valeur d'un système se construit avec ses utilisateurs. Cette idée nourrira plus tard les groupes d'utilisateurs, les interfaces ouvertes et les marchés OEM. Elle conduira aussi parfois **DEC** à sous-estimer l'avantage d'un logiciel plus fortement standardisé et contrôlé.

Surtout, le MIT donne au PDP-1 une histoire que les chiffres de vente n'auraient jamais produite. La machine devient associée à la culture hacker, aux outils interactifs, à *Spacewar!* et aux premières expériences de temps partagé. **DEC** acquiert la réputation d'un constructeur qui comprend les programmeurs et leur laisse une part d'initiative.

Le PDP-1 reste néanmoins coûteux et fabriqué en petit nombre. Pour transformer cette philosophie en véritable marché, **DEC** doit encore réduire le prix et les dimensions de ses machines. L'étape suivante ne consistera pas à perfectionner indéfiniment une architecture de 18 bits, mais à concevoir un ordinateur plus étroit, plus simple et beaucoup moins cher.

Chapitre 7

Le PDP-8 : douze bits pour changer d'échelle

Le PDP-1 montre qu'un ordinateur interactif peut séduire des chercheurs et des ingénieurs. Il ne résout cependant pas la question du nombre. À plus de 100,000 dollars, la machine reste réservée aux universités, aux centres de recherche et aux grandes organisations. Pour atteindre les laboratoires plus modestes, les ateliers et les fabricants d'instruments, **DEC** doit réduire le coût de façon radicale.

La réponse ne consiste pas à fabriquer un PDP-1 plus petit. Elle passe par une autre famille d'ordinateurs, fondée sur des mots de 12 bits et sur une architecture volontairement limitée. Cette voie commence avec le PDP-5, mais c'est le PDP-8 qui transforme véritablement l'échelle du marché.

7.1 Le PDP-5 ouvre la voie

En 1962, *Gordon Bell* commence à définir l'architecture du PDP-5 à partir d'un contrôleur 12 bits étudié auparavant par **DEC**. Il en fixe notamment le jeu d'instructions. Au début de l'année suivante, *Edson de Castro* poursuit le développement de la machine, annoncée en avril 1963.¹

Le PDP-5 introduit une organisation importante pour l'avenir des petits ordinateurs de **DEC**. Les périphériques ne sont plus raccordés uniquement par une structure conçue séparément pour chacun d'eux. Ils communiquent avec le processeur par l'intermédiaire d'un bus d'entrée-sortie plus général. Un constructeur peut ainsi ajouter un lecteur, un convertisseur analogique ou un dispositif

1. DIGITAL EQUIPMENT CORPORATION, INFORMATION RESEARCH SERVICES, cf. note 4.



Figure 7.1 – Le PDP-5 commercialisé par DEC à partir de 1963.

*Source : Bubba73. CC BY-SA 4.0, via Wikimedia Commons,
<https://commons.wikimedia.org/w/index.php?curid=179639623>.*

de commande sans redessiner l'ordinateur entier.

Ce changement intéresse particulièrement les laboratoires. Dans ces environnements, l'ordinateur n'est pas toujours utilisé pour effectuer de longs calculs. Il doit souvent recevoir une mesure, détecter un événement, commander un appareil ou enregistrer une série de résultats. La possibilité de connecter des équipements variés compte alors autant que la puissance du processeur.

Le choix de mots de 12 bits participe au même effort d'économie. Chaque bit supprimé réduit la largeur des registres, des circuits arithmétiques, des chemins de données et de la mémoire. La machine demande moins de composants, consomme moins d'énergie et occupe moins de place. En contrepartie, le programmeur doit composer avec des nombres plus courts, un espace d'adressage réduit et un jeu d'instructions minimal.

7.2 Une architecture dictée par le prix

Le PDP-8 reprend et simplifie l'architecture du PDP-5. Sa mémoire de base contient 4,096 mots de 12 bits. Elle peut être imaginée

comme une suite de 4,096 cases numérotées, chacune contenant une combinaison de douze zéros ou uns. La mémoire ne distingue pas les instructions des données : la signification d'un mot dépend de l'usage qu'en fait le processeur. La même suite de bits peut donc représenter une instruction, un nombre ou une adresse.

Une instruction doit elle aussi tenir dans un mot de 12 bits. Tous les bits doivent donc être partagés entre l'opération à exécuter et les informations nécessaires à cette opération. Pour les instructions qui accèdent à la mémoire, les trois bits de poids fort forment le code d'opération. Trois bits autorisent 2^3 , soit huit combinaisons : le PDP-8 ne dispose donc que de huit codes d'opération principaux.

Cette limitation ne résulte pas uniquement du choix des 12 bits. Les concepteurs auraient pu réserver davantage de bits au code d'opération, multiplier les formats ou employer des instructions composées de plusieurs mots. Ils auraient toutefois réduit la place disponible pour les adresses, accru le nombre d'accès à la mémoire et compliqué les circuits de décodage. **DEC** privilégie des instructions courtes et régulières, toutes contenues dans un seul mot.

Six codes concernent directement la mémoire : **AND**, **TAD**, **ISZ**, **DCA**, **JMS** et **JMP**. Les deux autres, **IOT** et **OPR**, servent respectivement aux entrées-sorties et aux opérations internes du processeur.

Ce petit nombre d'instructions oblige à combiner des opérations élémentaires. **TAD** additionne par exemple un mot à l'accumulateur. **DCA** dépose le contenu de celui-ci en mémoire, puis le remet à zéro. **ISZ** incrémente un mot et saute l'instruction suivante si le résultat devient nul. Une boucle peut ainsi être construite sans instruction particulière de comparaison.

L'adressage illustre directement les contraintes imposées par les mots de 12 bits. Dans une instruction qui accède à la mémoire, trois bits indiquent l'opération à effectuer. Un quatrième bit précise si l'adresse est directe ou indirecte, tandis qu'un cinquième sélectionne l'une des deux zones accessibles. Il ne reste donc que sept bits pour indiquer la position du mot recherché :

$$3 + 1 + 1 + 7 = 12.$$

Ces sept bits permettent de désigner 2^7 , soit 128 positions. Toutefois, ils ne suffisent pas à choisir directement un mot parmi

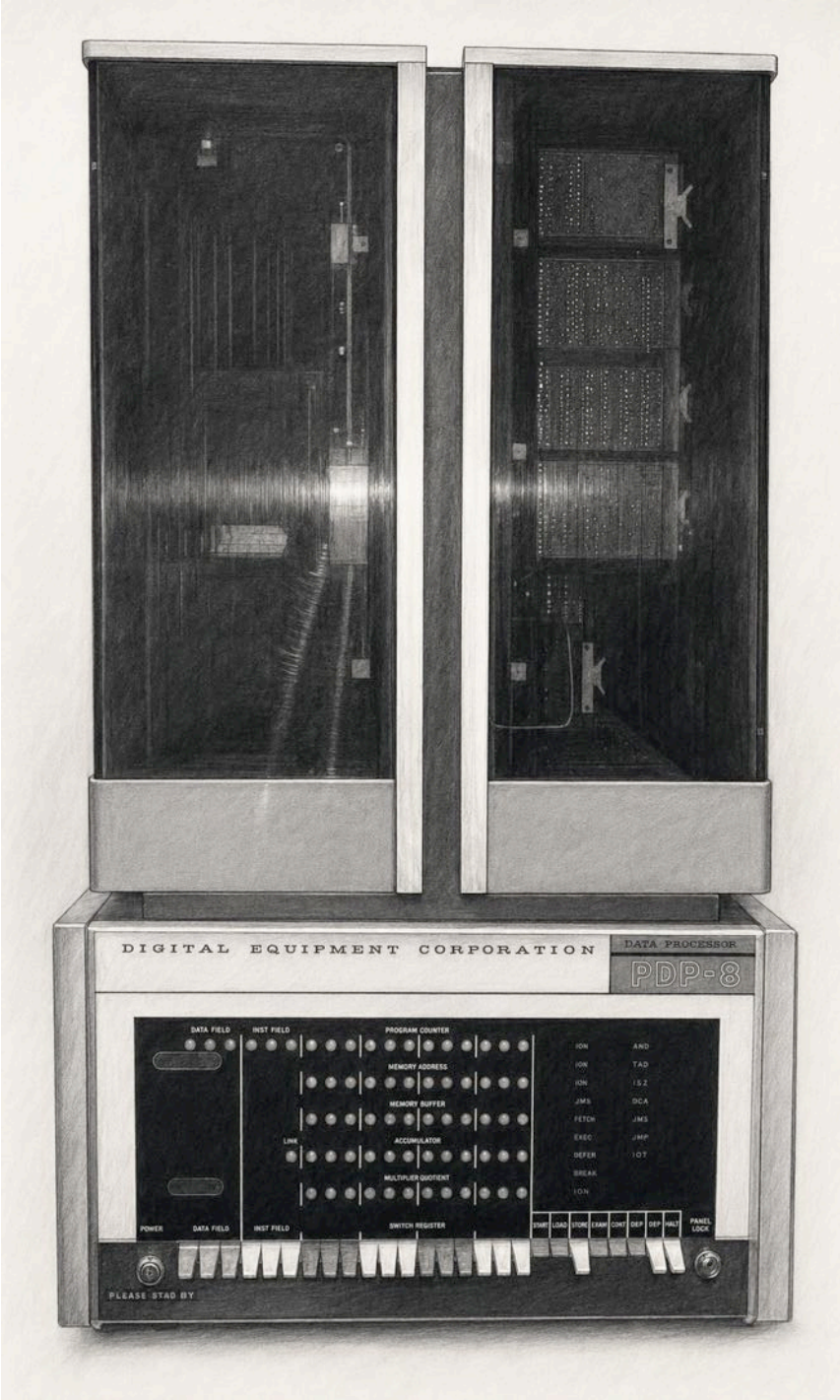


Figure 7.2 – Le PDP-8 dans sa configuration d’origine, avec ses modules électroniques disposés au-dessus du panneau de commande.

Source : Illustration réalisée avec OpenAI. D’après une photographie du domaine public, via Wikimedia Commons, <https://commons.wikimedia.org/w/index.php?curid=517876>.

les 4 096 que contient la mémoire de base. Le bit de page complète donc l'adresse. Selon sa valeur, les sept bits désignent soit un mot de la page zéro, formée des 128 premiers mots de la mémoire, soit un mot de la page dans laquelle se trouve l'instruction en cours.

Une instruction peut ainsi accéder directement à deux groupes de 128 mots : la page zéro et sa propre page. Cette organisation explique le rôle particulier de la page zéro, où les programmes placent fréquemment des constantes, des variables communes et des pointeurs.

Lorsqu'une donnée se trouve ailleurs, le programme peut employer l'adressage indirect. L'adresse inscrite dans l'instruction ne désigne alors pas la donnée elle-même, mais un mot mémoire qui contient son adresse complète sur 12 bits. Ce mot joue le rôle de pointeur. Le processeur doit effectuer un accès supplémentaire, mais il peut ainsi atteindre n'importe lequel des 4 096 mots de la mémoire de base.

Les extensions de mémoire ajoutent encore un niveau. Elles portent la capacité maximale à 32 768 mots, organisés en huit champs de 4 096 mots. Une adresse de 12 bits ne permet toujours de désigner qu'un mot à l'intérieur d'un seul champ. Des registres spécialisés indiquent donc au processeur dans quel champ se trouvent les instructions et les données. Une position complète ne dépend alors plus seulement d'une adresse comprise entre 0 et 4 095, mais aussi du champ auquel cette adresse appartient.

L'instruction **OPR** utilise autrement les bits disponibles. Son mot ne contient pas d'adresse : plusieurs bits commandent directement des opérations sur l'accumulateur ou sur le bit de retenue, appelé *Link*. Ils permettent notamment de les effacer, de les compléter, d'incrémenter l'accumulateur ou d'effectuer des rotations. Certaines de ces opérations peuvent être réunies dans une seule instruction. Le terme de micro-instruction employé dans la documentation désigne ces commandes élémentaires combinables, et non un microprogramme interne comparable à celui de processeurs plus complexes.

Tous ces choix poursuivent le même objectif. Des mots étroits, des instructions tenant dans un seul mot et un adressage limité réduisent le nombre de circuits nécessaires. En contrepartie, une

fonction absente du matériel doit être reconstruite par plusieurs instructions ou par un sous-programme. Une partie du coût passe ainsi du matériel vers le logiciel : le client achète moins de circuits, mais le programmeur doit davantage composer avec les limites de la machine.

7.3 Le PDP-8 change d'échelle

Présenté en avril 1965, le PDP-8 transforme en succès commercial les compromis techniques déjà expérimentés par **DEC**. Comme le PDP-7A contemporain, il utilise les nouveaux modules *Flip Chip*, développés par l'entreprise à partir de 1964. Plus petits que les cartes employées dans le PDP-5, ces modules enfichables portent les composants nécessaires aux circuits logiques de la machine.

Comme le montre la figure 7.3, le PDP-8 original est construit à partir de modules *Flip Chip* utilisant des composants discrets. Le module R211 représenté ici prend en charge un bit de trois registres du processeur. Douze modules semblables fonctionnent en parallèle pour former les registres de 12 bits de la machine.

Leur intérêt ne réside pas seulement dans leur taille. Les cartes sont insérées dans des connecteurs dont les broches dépassent à l'arrière du châssis. Pour relier électriquement ces broches, un fil dénudé est enroulé très serré autour de chacune d'elles selon la technique du *wire-wrap*. Une machine peut automatiser une partie de ce travail en suivant le plan de câblage prévu.

Comme le montre la figure 7.4, l'arrière du fond de panier présente un enchevêtrement de fils qui paraît désordonné. Ce câblage n'est pourtant pas improvisé : chaque broche est identifiée par la position de son connecteur et par un numéro. Pour chaque fil, le plan de fabrication indique précisément la broche de départ et celle d'arrivée. L'opérateur, ou la machine de câblage, exécute ainsi une liste de connexions déterminée à l'avance. Une fois le câblage terminé, des tests électriques vérifient la continuité de chaque liaison et permettent de repérer les connexions manquantes ou les courts-circuits.

Cette méthode accélère l'assemblage et réduit le risque d'erreur par rapport à un câblage entièrement réalisé à la main. Elle facilite

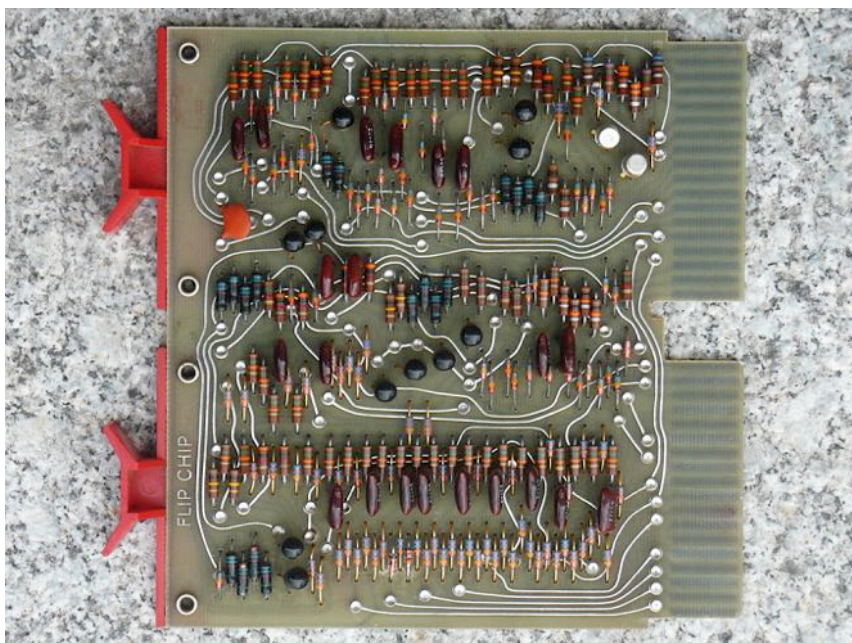


Figure 7.3 – Module *Flip Chip* R211 du PDP-8 original. Cette carte réalise une tranche d'un bit du compteur ordinal, du registre d'adresse mémoire et du registre de données mémoire.

Source : Douglas W. Jones. Photographie placée sous licence CC0 1.0, via Wikimedia Commons.

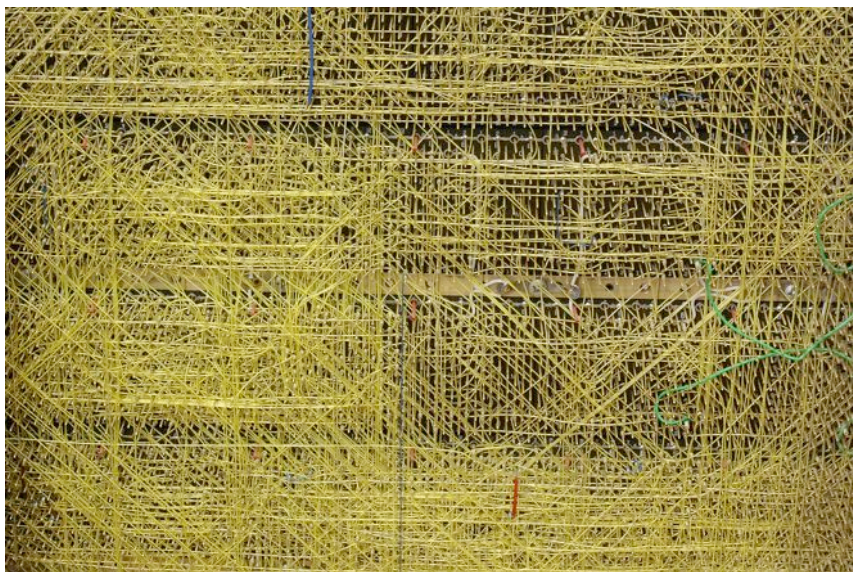


Figure 7.4 – Câblage par enroulement de fils à l’arrière du fond de panier d’un PDP-8/I. Les fils relient les broches des connecteurs dans lesquels sont insérées les cartes électroniques.

Source : Dave Fischer. CC BY-SA 3.0, <https://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons, <https://commons.wikimedia.org/wiki/File:PDP-8I-backplane.jpg>.

ainsi la fabrication du PDP-8 en série et contribue à diminuer son coût, même si elle n’est ni inventée pour cette machine ni réservée à celle-ci.

Doté de 4,096 mots de mémoire et accompagné d’un télétype ASR-33, le système est annoncé à environ 18,000 dollars. La somme reste considérable pour un particulier, mais elle entre dans le budget d’un laboratoire, d’un service d’ingénierie ou d’un fabricant d’équipements. Un ordinateur programmable n’est plus nécessairement un investissement réservé aux plus grandes institutions.

La réduction concerne aussi l’encombrement. Une photographie publicitaire montre un PDP-8 installé sur la banquette arrière d’une Volkswagen Coccinelle. La mise en scène est calculée : une machine que l’on peut transporter en voiture n’appartient plus à la même catégorie que les ordinateurs occupant une salle entière.

Le panneau avant conserve pourtant un rapport très direct avec le matériel. Ses commutateurs permettent d’introduire une adresse ou une valeur dans les registres. Après la mise sous tension,

l'opérateur saisit généralement quelques instructions constituant un programme de chargement, ou *Rim Loader*. Celui-ci lit ensuite un programme plus important depuis un ruban perforé ou un autre support.

L'opérateur ne saisit donc pas manuellement l'application entière. Il entre seulement, au moyen du panneau de commande, un court programme d'amorçage. Celui-ci permet de lire un chargeur plus élaboré, capable à son tour de charger un assembleur ou un programme d'application depuis un ruban perforé ou un autre périphérique. Sur les modèles ultérieurs équipés de disques, ce principe d'amorçage permet de lancer un système d'exploitation complet comme OS/8.

Parallèlement, l'environnement logiciel du PDP-8 s'enrichit. Il comprend plusieurs assembleurs, mais aussi des langages de haut niveau comme FORTRAN, BASIC et FOCAL², avec les compilateurs ou interpréteurs nécessaires à leur utilisation. Ces outils rendent la programmation plus accessible et permettent d'employer la machine dans des domaines très variés : acquisition de mesures, commande d'expériences, calcul scientifique ou enseignement de la programmation.

Le changement d'échelle ne vient donc pas seulement du prix affiché. DEC réunit une architecture économique, des périphériques utilisables et un ensemble croissant de logiciels. Le PDP-8 cesse d'être uniquement une machine plus petite : il devient un outil que des organisations beaucoup plus nombreuses peuvent acquérir, programmer et intégrer à leur travail.

7.4 Un ordinateur parfois invisible

Le PDP-8 n'est pas toujours installé comme un ordinateur autonome. Il peut être intégré à un analyseur scientifique, à un appareil médical, à une machine industrielle ou à un système d'affichage. Dans ce cas, l'utilisateur final ne voit parfois ni le panneau avant ni

2. Acronyme de *FO*rmula *CAL*culator, FOCAL est un langage de haut niveau interactif et interprété, développé en 1968 chez DEC par Richard Merrill pour le PDP-8. Il est principalement destiné au calcul scientifique et à l'enseignement de la programmation.

le nom de **DEC**. Il utilise un équipement spécialisé dont l'ordinateur constitue la partie programmable.

Ce marché des OEM devient l'un des moteurs du succès de la famille.³ Le fabricant connaît les capteurs, les procédés industriels ou les besoins médicaux de ses clients. **DEC** lui fournit une plateforme programmable, documentée et reproductible.

Cette répartition des rôles multiplie les applications sans obliger **DEC** à concevoir elle-même chaque produit final. Des PDP-8 sont employés pour analyser des signaux dans des laboratoires, surveiller des instruments dans des blocs opératoires, commander des tableaux d'affichage ou contrôler des installations scéniques.⁴

L'invisibilité de la machine constitue alors une réussite. Le client n'achète pas nécessairement un ordinateur : il achète un appareil qui mesure, affiche ou commande. Le PDP-8 devient un composant d'un système plus vaste, de la même manière qu'un microprocesseur le deviendra plus tard.

7.5 Une architecture qui traverse les générations

Le PDP-8 ne désigne bientôt plus un modèle unique, mais une famille de machines. Chaque nouvelle version cherche un équilibre différent entre vitesse, encombrement et prix, tout en conservant les principes essentiels de l'architecture d'origine.

Le PDP-8/S traite les opérations arithmétiques bit après bit. Cette organisation série réduit le nombre de circuits nécessaires et permet d'abaisser le prix de la machine, mais ralentit fortement son fonctionnement. Le PDP-8/I marque ensuite le passage aux circuits intégrés, tandis que le PDP-8/L reprend la même architecture dans une version plus compacte et moins coûteuse.

En 1970, le PDP-8/E réorganise la machine autour de l'OMNI-BUS. Ce bus commun assure les échanges entre le processeur, la mémoire et les contrôleurs de périphériques. Il simplifie la construction du système, facilite l'ajout de nouvelles interfaces et contribue

3. Un *OEM*, ou fabricant d'équipement d'origine, intègre le matériel d'un autre constructeur dans son propre produit, qu'il commercialise ensuite sous la forme d'un système complet.

4. COMPUTER HISTORY MUSEUM. *The PDP-8*.

à réduire les coûts de fabrication. Proposé initialement à 6 500 dollars, le PDP-8/E devient suffisamment abordable pour être intégré à des instruments scientifiques, des systèmes industriels ou des équipements spécialisés.⁵

Les technologies employées changent, mais l'organisation logique de la machine et l'essentiel de son jeu d'instructions sont préservés. Les programmes consacrés uniquement au calcul peuvent ainsi être transférés assez facilement d'un modèle à l'autre. La compatibilité devient toutefois moins certaine lorsqu'un logiciel dépend de la configuration de la mémoire, d'un périphérique particulier ou d'une interface propre à une installation. Malgré ces différences, les connaissances acquises sur une génération restent largement utilisables sur les suivantes.

Cette continuité favorise également l'enseignement. Les registres visibles par le programmeur sont peu nombreux et le jeu d'instructions peut être étudié dans son ensemble. Les échanges avec la mémoire et les périphériques demeurent suffisamment simples pour que l'on puisse suivre les opérations réellement exécutées.

Les limites de la machine deviennent alors une qualité pédagogique. Elles obligent à comprendre comment une adresse est formée, comment une donnée circule et comment une instruction agit sur l'état du processeur. Le PDP-8 constitue ainsi non seulement un produit industriel durable, mais aussi une remarquable école d'architecture informatique.

7.6 Le mini-ordinateur devient un marché

Au cours de la seconde moitié des années 1960, le terme « mini-ordinateur » s'impose pour désigner des machines moins coûteuses et moins encombrantes que les grands systèmes. Elles peuvent être installées dans un laboratoire, affectées à un service particulier ou intégrées à un équipement scientifique ou industriel.

Le PDP-8 n'est ni le premier petit ordinateur ni le seul présent sur ce marché. Il en devient néanmoins le modèle emblématique. Pendant environ vingt-cinq ans, **DEC** en développe plus d'une

5. COMPUTER HISTORY MUSEUM, *The PDP-8*, cf. note 4.

douzaine de variantes et en vend plus de 10 000. Cette diffusion montre que l'ordinateur n'est plus nécessairement réservé à quelques centres disposant de moyens exceptionnels. Il peut désormais être acquis pour répondre à un besoin local et clairement défini.⁶

Le succès de cette formule attire rapidement des concurrents. En 1968, *Edson de Castro*, qui avait participé à la conception du PDP-8, quitte **DEC** et cofonde **Data General**. Le Nova, premier ordinateur de la nouvelle société, adopte une architecture 16 bits tout en conservant une construction compacte et un prix compétitif. Il vise directement les laboratoires, les industriels et les intégrateurs qui constituent une part importante de la clientèle de **DEC**.

L'apparition de Data General ne provoque pas à elle seule la naissance du PDP-11. **DEC** étudiait déjà plusieurs projets de machines 16 bits. Elle rend toutefois plus urgente la recherche d'un successeur capable de dépasser les contraintes de l'architecture 12 bits. Le futur PDP-11 devra conserver la souplesse et le coût raisonnable des petits systèmes, tout en proposant des registres plus nombreux, un adressage plus régulier et une architecture plus facile à programmer.

Le succès du PDP-8 révèle enfin qu'une unité centrale ne suffit pas à former un système réellement utilisable. Selon leur activité, les clients ont également besoin de télétypes, de terminaux d'affichage, de lecteurs de bande, de disques, d'interfaces analogiques ou numériques et de logiciels adaptés.

À mesure que ces configurations se développent, **DEC** ne vend plus seulement de petits ordinateurs. L'entreprise doit fournir les périphériques, les interfaces et les logiciels qui permettent de les relier au monde réel. Pour comprendre cette transformation en constructeur de systèmes complets, il faut maintenant suivre les câbles qui partent de la machine.

6. COMPUTER HISTORY MUSEUM, *The PDP-8*, cf. note 4.

Troisième partie

Du laboratoire au marché

Chapitre 8

Brancher le monde réel

Un ordinateur n'entre pas dans un laboratoire uniquement par son processeur. Il doit recevoir des mesures, commander des appareils, conserver des programmes et restituer des résultats. Dans l'histoire de **DEC**, les périphériques ne sont donc pas des accessoires ajoutés après la conception. Ils déterminent les lieux où la machine peut travailler et les personnes qui peuvent l'utiliser.

Le PDP-1 avait déjà montré l'importance d'une structure d'entrée-sortie ouverte. Le PDP-8 généralise cette relation au monde extérieur. Un fabricant d'instruments peut relier des convertisseurs analogiques, des compteurs, des relais ou un dispositif d'affichage. Le manuel d'interface devient presque aussi important que le manuel de programmation : il explique comment faire entrer un phénomène physique dans la logique de la machine.

L'ouverture matérielle n'est utile que si elle est documentée. Les manuels d'interface de **DEC** décrivent les niveaux de tension, les temporisations, les signaux de commande et les procédures d'interruption. Pour un ingénieur en instrumentation, ces pages comptent davantage qu'une présentation générale du processeur. Elles lui permettent de savoir si son appareil pourra imposer son rythme à la machine et comment éviter de perdre une mesure.

Cette documentation encourage une industrie périphérique. Des entreprises spécialisées proposent des convertisseurs, des multiplexeurs, des contrôleurs et des châssis. Elles enrichissent les systèmes sans que **DEC** doive devenir experte de chaque discipline. Le bus ou l'interface agit alors comme un contrat technique : chacun peut innover de son côté à condition de respecter les signaux communs.

8.1 Le télétype, porte d'entrée ordinaire

Le Teletype Model 33 ASR accompagne de nombreuses petites installations informatiques des années 1960 et 1970. Il réunit un clavier, une imprimante, un lecteur et un perforateur de ruban papier. Il ne réalise lui-même aucun calcul : il permet à l'utilisateur de communiquer avec l'ordinateur et de conserver des programmes sur un support matériel.



Figure 8.1 – Télétype Model 33 ASR équipé d'un lecteur et d'un perforateur de ruban papier.

Source : Illustration réalisée avec OpenAI d'après une photographie de TedColes. Wikimedia Commons, licence CC BY-SA 4.0.

Sur un PDP-8, le télétype est relié par un câble à une interface installée dans la machine. La communication s'effectue en série, caractère après caractère. Dans la configuration habituelle, le clavier et le lecteur de ruban constituent la voie d'entrée, tandis que l'imprimante et le perforateur forment la voie de sortie.

Lorsqu'une touche est enfoncée, le Model 33 transmet le code du caractère à l'interface. Celle-ci le conserve jusqu'à ce que le programme exécuté par le PDP-8 vienne le lire. Dans l'autre sens,

l'ordinateur envoie un caractère à l'interface de sortie ; le télétype le reçoit et l'imprime sur le papier. La réponse visible de la machine est donc produite par une succession d'échanges très simples.

Une fois un interpréteur, un éditeur ou un système chargé, l'utilisation devient assez directe. Le programme imprime un message d'attente ou une invite, l'utilisateur saisit une commande, puis appuie sur la touche de retour du chariot. Le logiciel analyse alors la ligne et imprime sa réponse. Avec BASIC, par exemple, l'utilisateur peut saisir une instruction, lancer son programme, puis lire les résultats sur le rouleau de papier.

La correction d'une faute dépend du logiciel, et non du télétype seul. L'imprimante ne peut pas revenir sur le papier pour faire disparaître un caractère déjà frappé. Dans l'éditeur symbolique du PDP-8, la touche RUB OUT supprime cependant le dernier caractère conservé dans la mémoire de l'ordinateur. L'éditeur imprime une barre oblique inverse pour indiquer cette suppression.

Supposons que l'utilisateur veuille écrire PRINT, mais saisisse PRNIT. Il peut appuyer deux fois sur RUB OUT, puis taper les trois derniers caractères correctement. Le papier porte alors une trace semblable à celle-ci :

PRNIT\\INT

Les deux barres indiquent que les deux caractères précédents ont été annulés. Dans la mémoire de l'éditeur, seule la forme correcte PRINT est conservée. Une autre touche, marquée d'une flèche vers la gauche, permet d'annuler toute la ligne en cours. Si la ligne a déjà été validée, l'utilisateur doit employer les commandes de l'éditeur pour l'afficher, la remplacer ou la supprimer. Le fonctionnement exact varie cependant selon l'interpréteur ou l'éditeur utilisé.

Le lecteur de ruban fournit ses caractères par le même chemin que le clavier. L'utilisateur place le ruban dans le mécanisme, puis lance sa lecture. Les rangées de perforations sont converties successivement en caractères et transmises à l'ordinateur. Dans l'autre sens, le logiciel peut envoyer un texte vers le perforateur afin d'en produire une copie sur un ruban neuf.

Lorsqu'une faute se trouve déjà enregistrée sur un ruban, elle est plus difficile à corriger. Le papier perforé ne se modifie pas aussi

facilement qu'un texte conservé en mémoire. La méthode la plus sûre consiste à lire le ruban avec un éditeur, à corriger le contenu dans la mémoire du PDP-8, puis à perforer un nouveau ruban. Dans certains cas, un caractère pouvait aussi être neutralisé en le transformant en code RUB OUT, que le logiciel ignorait lors de la lecture.

Au démarrage, comme nous l'avons vu, le lecteur peut d'abord servir à charger le *Binary Loader* au moyen du RIM Loader, le petit programme introduit depuis le panneau de commande. Grâce à la mémoire à tores magnétiques, celui-ci reste généralement disponible après l'extinction de la machine, tant que sa zone mémoire n'a pas été écrasée. Une fois le chargeur lancé, les programmes enregistrés sur ruban peuvent être introduits directement depuis le télétape.

Avec une vitesse d'environ dix caractères par seconde, une longue impression demande du temps. Les programmeurs privilégient donc les commandes courtes, les messages concis et les éditeurs travaillant ligne par ligne. Le bruit de l'impression, les corrections encore visibles et le défilement du papier font ainsi partie intégrante du dialogue avec la machine.

8.2 DECtape

Le nom DECtape ne désigne pas un lecteur particulier. Il s'applique d'abord au support magnétique et à son format d'enregistrement. Le ruban, large d'environ 19 mm, est enroulé sur de petites bobines transportables. Pour l'utiliser, il faut un lecteur, également appelé *transport*, ainsi qu'un contrôleur assurant les échanges avec l'ordinateur.

Le système trouve son origine dans les recherches menées au Lincoln Laboratory sur les bandes magnétiques à accès rapide, puis dans le LINCtape employé par le LINC. En 1963, **DEC** commercialise une première version sous le nom de *Microtape*, composée du transport double Type 555 et du contrôleur Type 550. Le nom DECtape apparaît à partir de 1964. Les transports porteront ensuite notamment les références TU55 et TU56.¹

1. DIGITAL EQUIPMENT CORPORATION, INFORMATION RESEARCH SERVICES, cf. note 4.



Figure 8.2 – Transport double TU56 pour rubans DECTape, ici associé à un PDP-11.

Source : Photographie de ndiggity. Wikimedia Commons, licence CC BY-SA 3.0.

La différence essentielle avec une bande magnétique traditionnelle réside dans l'organisation du support. Un DECTape est préparé par une opération de formatage qui inscrit des repères permanents et le divise en blocs numérotés. Chaque bloc possède ainsi une position logique connue, même si sa distance physique par rapport aux têtes de lecture varie selon l'endroit où le ruban s'est arrêté.

La machine peut donc rechercher un bloc déterminé sans transmettre au programme toutes les données qui le précèdent. Le ruban défile dans un sens ou dans l'autre, tandis que les numéros des blocs passent devant les têtes. Lorsque le numéro attendu est rencontré, la lecture ou l'écriture des données peut commencer.

Le DECTape reste néanmoins une bande. L'accès n'est pas instantané comme sur un disque : si le bloc recherché se trouve loin de la position courante, les bobines doivent tourner jusqu'à ce qu'il atteigne les têtes. Du point de vue du logiciel, le support se comporte cependant comme un disque lent, puisque chaque bloc peut être retrouvé, lu et réécrit séparément.

Sur un PDP-8, un ruban comporte habituellement 1,474 blocs

physiques de 129 mots de 12 bits. OS/8 n'utilise que les 128 premiers mots de chacun de ces blocs, soit une capacité utile d'environ 184 K mots de 12 bits, ou 276 Kio. Il regroupe ensuite deux blocs physiques successifs pour former un bloc logique de 256 mots, unité employée par son système de fichiers.

Le transport constitue la partie électromécanique du dispositif. Pour chaque unité, deux moteurs entraînent directement les moyeux sur lesquels sont placées la bobine de DECTape et la bobine réceptrice. Des guides maintiennent le ruban devant une tête chargée de lire ou d'inscrire les signaux magnétiques. Le transport assure ainsi le déplacement du ruban, mais il ne connaît ni les noms des fichiers ni leur organisation.

Il est relié à un contrôleur adapté à la famille d'ordinateurs utilisée, par exemple le TC08 sur certaines configurations PDP-8 ou le TC11 sur le PDP-11. Un pilote logiciel commande à son tour ce contrôleur.

Pour retrouver les fichiers, OS/8 maintient un répertoire enregistré sur le DECTape lui-même. Ce répertoire joue le rôle d'une table des matières : il permet au système de connaître le nom, la taille et l'emplacement logique de chaque fichier. OS/8 lui réserve les blocs logiques 1 à 6, dont la position est fixée par l'organisation du système.

La simple installation d'une bobine ne provoque pas nécessairement la lecture de tout le répertoire. OS/8 le consulte lorsqu'une commande doit afficher le contenu du DECTape, ouvrir un fichier ou lancer un programme. Si l'utilisateur demande, par exemple, l'exécution de **BASIC.SV**, le système recherche son nom dans le répertoire et détermine le numéro de son premier bloc ainsi que sa longueur.

Le démarrage depuis un DECTape constitue un cas particulier. OS/8 n'a pas besoin de consulter le répertoire pour trouver son premier programme, car le bloc logique 0 est réservé à l'amorçage. Le petit programme lancé depuis le PDP-8 recherche directement ce bloc, qui introduit les premiers éléments du système. Une fois OS/8 actif, le répertoire lui permet de retrouver les autres programmes à partir de leur nom.

Lorsqu'un fichier doit être chargé, le pilote connaît donc son

numéro de bloc, mais pas sa position physique en centimètres sur la bande. Il commande le déplacement du ruban et examine les numéros détectés par le contrôleur. Si le numéro rencontré indique que le bloc recherché se trouve dans l'autre direction, le mouvement est inversé. Lorsque le bon numéro atteint les têtes, le contrôleur transfère le contenu du bloc entre le transport et la mémoire du PDP-8. Les fichiers d'OS/8 occupant des blocs consécutifs, les blocs suivants peuvent ensuite être lus dans l'ordre.

Concrètement, l'utilisateur installe une bobine, engage le ruban sur la bobine réceptrice, sélectionne l'unité et la place en service. Il peut alors demander au système d'afficher le répertoire, de charger un programme, d'ouvrir un fichier ou d'enregistrer des données. Une seule bobine peut contenir le système, ses utilitaires, plusieurs programmes et les fichiers de travail d'un utilisateur.

La fiabilité du support repose notamment sur ses dix pistes physiques, regroupées en cinq canaux redondants : un canal de synchronisation, un canal de marquage et trois canaux destinés aux données. Chaque canal est enregistré sur deux pistes non adjacentes. Une détérioration localisée d'une partie du ruban peut ainsi laisser intacte la seconde piste portant la même information. Les repères de synchronisation et de marquage, inscrits lors du formatage, permettent également de reconnaître les limites et les numéros des blocs.

DECtape occupe ainsi une position intermédiaire entre le ruban perforé et le disque. Il est réutilisable, transportable et suffisamment robuste pour conserver des fichiers de travail, alors que les disques restent encore coûteux. La modification d'un fichier n'impose pas de réécrire l'ensemble de la bande, et certaines petites installations peuvent même faire fonctionner leur système directement depuis DECtape.

Les mouvements des TU55 et TU56 deviennent enfin l'une des images familières des systèmes **DEC**. Pendant la recherche d'un bloc, les bobines accélèrent, ralentissent et changent parfois brusquement de direction. Cette activité visible rappelle que, malgré une organisation logique proche de celle d'un disque, le temps d'accès dépend toujours du déplacement mécanique du ruban.

8.3 Du disque au système de stockage

À mesure que les logiciels grandissent, le disque devient indispensable. Les familles RK, RL, RP puis RA couvrent des besoins allant du petit disque amovible aux ensembles de grande capacité. Le disque ne sert pas seulement à conserver davantage de données. Son accès direct transforme la conception des systèmes d'exploitation : répertoires, fichiers interactifs, échanges entre utilisateurs et bases de données deviennent plus pratiques.



Figure 8.3 – Deux unités de disques RP06 de **DEC**. Apparu dans les années 1970, ce modèle appartient à la famille RP, destinée aux systèmes nécessitant une capacité de stockage importante.

Source : Photographie de Joe Mabel. Wikimedia Commons, licence CC BY-SA 3.0.

Les supports amovibles permettent aussi de séparer les environnements. Un laboratoire peut conserver ses propres plateaux, un service changer de jeu de données et un technicien démarrer un système de diagnostic. Mais cette flexibilité a un prix : têtes de lecture, alignement, poussière et contrôleurs ajoutent de nouveaux points de panne. Le service de terrain de **DEC** devient alors une partie essentielle du produit.

Dans les années 1980, les contrôleurs HSC et les VAXclusters feront du stockage un service partagé entre plusieurs ordinateurs.

Cette évolution sera développée au chapitre 15. Elle commence pourtant avec une idée simple : le support de données doit être pensé avec la machine, non acheté comme une réflexion secondaire.

Le disque change l'échelle des logiciels. Un moniteur compact chargé depuis ruban peut se contenter de quelques commandes. Avec un accès direct plus rapide et une capacité croissante, le système peut maintenir des répertoires, conserver plusieurs versions et servir plusieurs utilisateurs. Le périphérique crée donc de nouvelles attentes qui reviennent vers le processeur : davantage de mémoire, de protection et de débit.

Les contrôleurs deviennent des ordinateurs spécialisés. Ils gèrent le déplacement des têtes, la détection des erreurs et les transferts de blocs. Cette intelligence déportée soulage le processeur principal, mais complexifie le diagnostic. Lorsque les données ne sont pas lues, la panne peut se trouver dans le support, l'électronique du lecteur, le câble, le contrôleur, le bus ou le logiciel. Le technicien de terrain doit comprendre toute la chaîne.

8.4 Voir, mesurer, agir

Les écrans graphiques donnent aux PDP une présence particulière. Le Type 30 du PDP-1 trace des points commandés directement par le programme. Les systèmes suivants ajoutent des possibilités de dessin, de conversion et d'interaction au crayon lumineux. Dans les laboratoires, l'affichage permet de suivre une expérience, de visualiser un signal ou de manipuler une représentation avant que le terme « interface graphique » ne devienne courant.

Les convertisseurs analogique-numérique accomplissent le trajet inverse : ils transforment une tension issue d'un capteur en valeur exploitable par le programme. Des convertisseurs numérique-analogique, des relais et des lignes de commande permettent ensuite d'agir sur l'expérience. Le mini-ordinateur devient ainsi le centre d'une boucle : mesurer, calculer, afficher, corriger.

Cette capacité explique la présence des PDP dans la médecine, la physique, la chimie, l'automatisation et le contrôle industriel. Elle explique aussi l'importance des fabricants tiers. Grâce aux bus et aux interfaces documentées, d'autres sociétés proposent des



Figure 8.4 – Precision CRT Display Type 30 de **DEC**, présenté avec son meuble électronique. Son tube cathodique circulaire de 30 pouces était destiné à l’affichage graphique de grande précision.

Source : Illustration réalisée d’après une photographie de Digital Equipment Corporation. Precision CRT Display Type 30, brochure F-13 (30), Digital Equipment Corporation, mai 1964.

cartes et des périphériques spécialisés. L’écosystème augmente la valeur de la machine sans que **DEC** doive tout concevoir.

Dans une expérience scientifique, la valeur du système se juge souvent à sa capacité à ne pas manquer l’événement. Un convertisseur peut produire des mesures plus vite que le programme ne les traite. Il faut alors des interruptions, des mémoires tampons ou des transferts directs. Ces mécanismes, que nous retrouverons avec l’Unibus du PDP-11, montrent comment un besoin physique pousse l’architecture à évoluer.

Le périphérique rapproche aussi **DEC** de métiers éloignés de l’informatique. Un médecin parle de signaux biologiques, un chimiste de spectres, un industriel de cycles de machine. L’ingénieur d’application doit traduire ces phénomènes en débits, résolutions et délais. L’entreprise grandit donc en apprenant le vocabulaire de ses clients.

Le succès des périphériques pose cependant une nouvelle question. Lorsqu’un ordinateur possède des disques, plusieurs terminaux et de

nombreux utilisateurs, il ne peut plus être seulement un instrument individuel. Il devient un lieu partagé. Les machines 36 bits vont explorer cette autre forme de proximité : non plus un utilisateur devant un processeur, mais une communauté entière connectée au même système.

Chapitre 9

Partager la puissance

Le PDP-8 rend l'ordinateur plus accessible en réduisant son prix et sa taille. Le PDP-6 suit une direction presque opposée. Annoncé en 1964, il utilise des mots de 36 bits, vise le calcul scientifique avancé et doit servir plusieurs utilisateurs à la fois. Pour **DEC**, entrer sur ce marché signifie affronter des machines plus puissantes tout en conservant l'interactivité qui distingue la société.



Figure 9.1 – Gordon Bell et Alan Kotok devant un PDP-6, vers 1964. Cette machine de 36 bits marque l'entrée de **DEC** dans le domaine des grands systèmes.

Source : Photographie du Computer History Museum. Wikimedia Commons, licence CC BY 3.0.

Seulement vingt-six PDP-6 sont installés selon la chronologie interne de **DEC**. Le modèle ne devient pas un succès commercial comparable au PDP-8. Il fournit cependant l'architecture et l'ex-

périence qui conduiront au PDP-10, l'une des machines les plus influentes des universités, des centres de recherche et des premiers réseaux.

La largeur de 36 bits n'est pas choisie pour impressionner. Elle convient aux calculs scientifiques en précision élevée et à la représentation de pointeurs ou de caractères groupés dans un mot. Elle rapproche également la machine des habitudes de certaines grandes installations universitaires. **DEC** veut montrer qu'elle peut fournir la puissance attendue sans renoncer à une console interactive et à une architecture compréhensible.

Le projet dépasse cependant les moyens habituels de la société. Les circuits, la mémoire et les logiciels coûtent cher, tandis que la clientèle reste plus étroite que celle du PDP-8. Le PDP-6 devient une expérience exigeante : techniquement féconde, commercialement fragile. Son importance vient précisément du fait que l'équipe ne l'abandonne pas après les faibles volumes. Elle transforme l'architecture en une famille mieux positionnée.

9.1 Une grande machine selon DEC

Le PDP-6 n'est pas un mini-ordinateur au sens habituel. Sa mémoire, ses périphériques et son coût le placent dans une autre catégorie. Mais son usage diffère du grand système de traitement par lots. Il est conçu pour *le temps partagé*, avec des terminaux à partir desquels les utilisateurs peuvent éditer, compiler et exécuter leurs programmes.

Gordon Bell dirige l'architecture initiale. *Alan Kotok*, arrivé du MIT, joue ensuite un rôle majeur dans la lignée. Les concepteurs connaissent directement les attentes des communautés interactives : réponse rapide, fichiers personnels, outils de débogage et possibilité de faire cohabiter calcul, édition et communications.

En novembre 1965, une démonstration spectaculaire illustre les possibilités du PDP-6. Depuis le Massachusetts, Alan Kotok et ses collègues commandent par télex une machine installée à l'université d'Australie-Occidentale, à Perth, à près de 19 000 km de distance. La liaison utilise un code de téléimprimeur dans lequel chaque caractère est représenté par cinq signaux binaires successifs.

Chacun pouvant prendre deux valeurs, ces cinq signaux permettent de former 2^5 , soit seulement 32 combinaisons différentes. Pour représenter malgré tout les lettres, les chiffres et les signes, le code utilise des commandes permettant de basculer entre plusieurs jeux de caractères. Conçu pour les télex, il ne prévoit cependant pas certaines commandes propres aux terminaux informatiques. L'envoi de l'équivalent de `Ctrl-C` pose notamment problème.

Au cours de la session, Robin Frith, présent à Perth, lance avec humour à Alan Kotok : « Pensez-vous pouvoir nous laisser, à nous autres pauvres Australiens, un peu de mémoire à tores ? » La plaisanterie prend tout son sens dans un système en temps partagé : bien que situé à proximité de la machine, l'utilisateur australien en partage les ressources avec des opérateurs intervenant depuis l'autre côté du monde. L'expérience montre ainsi qu'un ordinateur peut être utilisé à distance comme une ressource commune, malgré les limites encore importantes des télécommunications.¹

Le temps partagé modifie la perception de la puissance. Un utilisateur ne demande pas que le processeur lui appartienne en permanence ; il demande que le système réponde assez vite pour maintenir son attention. Une machine peut donc servir des dizaines de personnes si le moniteur répartit correctement le temps, protège les fichiers et déplace les travaux longs vers les moments disponibles.

Cette organisation produit un nouvel espace de travail. Le terminal n'affiche pas seulement un résultat : il donne accès à un répertoire, un éditeur, un compilateur et parfois aux messages des autres utilisateurs. La communauté se forme à l'intérieur du système. Les noms de fichiers, les conventions de commandes et les programmes partagés deviennent une culture locale.

Les machines 36 bits accueillent ainsi des pratiques dont l'influence dépassera leurs ventes : intelligence artificielle, traitement symbolique, édition interactive, courrier électronique et premiers jeux multiutilisateurs. Elles sont suffisamment puissantes pour faire fonctionner ces environnements et suffisamment ouvertes pour que les universités les transforment.

1. DIGITAL EQUIPMENT CORPORATION. *36-bit Systems Timeline*. Digital Computing History Timeline. 1997.

9.2 Le PDP-10 et ses communautés

Lorsque **DEC** commercialise le PDP-10 en 1967, la société ne conçoit pas une architecture entièrement nouvelle. La première version de la machine, équipée de l'unité centrale KA10, conserve les mots de 36 bits et l'essentiel du jeu d'instructions du PDP-6. Une grande partie des programmes existants peut donc être réutilisée, ce qui représente un avantage important pour les universités ayant déjà constitué leurs propres bibliothèques logicielles.

Le PDP-10 transforme ainsi l'expérience encore limitée du PDP-6 en une famille durable. Quatre générations d'unités centrales se succèdent : KA10, KI10, KL10 puis KS10. Elles exécutent largement les mêmes programmes, mais profitent des progrès rapides de l'électronique.

Le KA10, livré à partir de 1967, constitue la première réalisation de la famille. Encore construit à partir de transistors et de diodes, il reprend largement la conception du PDP-6 dans une organisation plus compacte et plus facile à fabriquer. Il permet ainsi à **DEC** de commercialiser plus largement son architecture 36 bits.

Le KI10, apparu en 1972, utilise des circuits intégrés et offre de meilleures performances. Il améliore surtout la gestion de la mémoire grâce à la pagination : la mémoire d'un programme est divisée en blocs appelés *pages*, que le système peut placer et protéger séparément. Ce mécanisme facilite le partage de la machine entre de nombreux utilisateurs.

Le KL10, commercialisé à partir de 1975, représente la version la plus puissante du PDP-10. Des circuits plus rapides et une petite mémoire cache, placée près de l'unité centrale, réduisent le temps nécessaire pour accéder aux instructions et aux données les plus souvent utilisées. Le KL10 équipe les grandes installations fonctionnant sous TOPS-10 ou TOPS-20.

Le KS10, lancé en 1978, poursuit un autre objectif. Il n'est pas plus puissant que le KL10, mais il est plus compact et moins coûteux. Les progrès de l'intégration électronique permettent de réaliser la même architecture avec moins de composants. Commercialisé sous le nom de DECSYSTEM-2020, il rend ainsi les systèmes 36 bits accessibles à des établissements disposant de moyens plus limités.

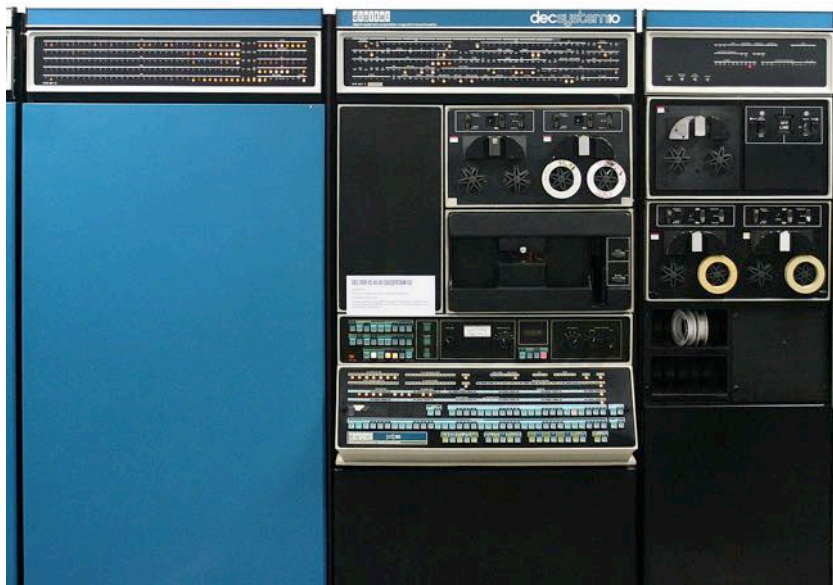


Figure 9.2 – Un PDP-10 équipé d’une unité centrale KI10. Cette machine de 36 bits appartient à la génération qui prolonge l’architecture inaugurée par le PDP-6.

Source : Photographie de Gah4, détourage de Pittigrilli. Wikimedia Commons, licence CC BY-SA 4.0.

Ces quatre unités centrales illustrent les transformations de l’informatique durant les années 1960 et 1970. Le KA10 rend l’architecture du PDP-6 plus facile à commercialiser, le KI10 améliore le temps partagé, le KL10 recherche la puissance et le KS10 réduit le coût et l’encombrement. Malgré ces évolutions, la compatibilité entre les différentes générations permet à de nombreux programmes de continuer à fonctionner.

9.2.1 Un environnement partagé

L’importance du PDP-10 ne réside pas seulement dans son processeur. La machine est conçue pour le temps partagé : plusieurs personnes, installées devant des terminaux différents, utilisent simultanément le même ordinateur. Le système répartit entre elles le temps du processeur, la mémoire et les périphériques. Chacune dispose d’une session, de fichiers et de programmes qui lui donnent l’impression de travailler sur une machine disponible pour elle seule.

TOPS-10 prolonge directement le moniteur conçu pour le PDP-6. Au fil de ses versions, il devient un environnement multiutilisateur complet, capable de gérer les comptes, les fichiers, les compilateurs, les terminaux, les travaux différés et les communications. L'utilisateur peut écrire un programme, le compiler, l'exécuter et le corriger depuis son terminal, sans préparer un paquet de cartes ni attendre le passage d'un opérateur.

Tous les établissements n'utilisent cependant pas le même système. Le laboratoire d'intelligence artificielle du MIT développe ITS, l'*Incompatible Timesharing System*, tandis que le laboratoire d'intelligence artificielle de Stanford emploie WAITS. Ces systèmes reflètent les habitudes de leurs communautés. Les utilisateurs ne se contentent pas des logiciels fournis par **DEC** : ils modifient leur environnement, construisent des commandes, des éditeurs et des langages, puis partagent leurs créations.

Le centre informatique devient dès lors un véritable espace social. Les utilisateurs consultent les fichiers des projets communs, échangent des messages et améliorent les outils dont ils se servent quotidiennement. Une partie de la culture des hackers se développe dans ce contexte, autour de l'idée qu'un programme peut être étudié, transformé et perfectionné collectivement.

Le PDP-10 accueille notamment MACLISP et le système de calcul symbolique Macsyma, qui jouent un rôle important dans les recherches en intelligence artificielle. En 1976, Richard Stallman et Guy Steele réunissent plusieurs ensembles de macros écrites pour l'éditeur TECO afin de former la première version d'Emacs. Celle-ci fonctionne sous ITS sur les PDP-10 du laboratoire d'intelligence artificielle du MIT. Le jeu d'aventure textuel *Zork*, développé quelques années plus tard sur la même famille de machines, naît lui aussi dans cette culture de programmation collective.

9.2.2 Du terminal au réseau

Cette communauté dépasse progressivement les limites du bâtiment. Plusieurs PDP-10 deviennent des hôtes importants de l'ARPANET. Un utilisateur peut ouvrir une session sur une machine éloignée, transférer des fichiers ou employer un programme

installé dans un autre établissement. Le temps partagé permettait déjà à plusieurs personnes d'utiliser le même ordinateur ; le réseau permet désormais à des communautés éloignées d'accéder aux mêmes ressources.

Le système TENEX, commencé en 1969 chez Bolt, Beranek and Newman, joue un rôle majeur dans cette évolution. Il fonctionne d'abord sur un KA10 modifié auquel BBN ajoute un mécanisme de pagination. Grâce à la mémoire virtuelle, chaque programme peut disposer d'un espace d'adressage qui n'a pas besoin d'être entièrement présent dans la mémoire physique. TENEX propose également un système de fichiers et un environnement interactif particulièrement élaborés pour l'époque.

En 1971, Ray Tomlinson utilise deux PDP-10 de BBN exécutant TENEX pour expérimenter l'envoi d'un message entre deux ordinateurs reliés à l'ARPANET. Il adapte le programme local **SNDMSG** et choisit le caractère @ pour séparer le nom de l'utilisateur de celui de la machine destinataire. Le courrier électronique existait déjà à l'intérieur de certains systèmes en temps partagé, mais il franchit alors la frontière séparant deux ordinateurs.

DEC reconnaît finalement la valeur de TENEX et s'en inspire pour créer TOPS-20, commercialisé en 1976 avec le DECSYSTEM-20. Malgré son nom, TOPS-20 n'est donc pas simplement la version suivante de TOPS-10 : il appartient à une autre lignée logicielle. TOPS-10 prolonge le moniteur du PDP-6, tandis que TOPS-20 hérite de la mémoire virtuelle et de l'environnement interactif développés par BBN. Deux cultures logicielles coexistent ainsi au sein de la même famille de machines.

9.2.3 Un héritage durable

Le PDP-10 occupe une place particulière dans l'histoire de l'informatique. Il n'impose pas un modèle unique, mais fournit à plusieurs communautés un espace dans lequel elles peuvent expérimenter de nouvelles manières de programmer, de travailler ensemble et de communiquer. Le temps partagé, les langages issus de Lisp, les premiers environnements d'Emacs, les jeux d'aventure textuels et les échanges sur l'ARPANET se développent dans cette culture

interactive.

Lorsque **DEC** met fin à sa famille de machines 36 bits en 1983, remplacer le matériel ne suffit donc pas. Chaque centre possède des comptes, des fichiers, des programmes locaux, des outils adaptés à ses besoins et parfois plusieurs années de correspondance. Il ne s'agit pas seulement de transférer des données vers une nouvelle machine, mais de reconstruire tout un environnement de travail.

La disparition du PDP-10 est ainsi ressentie par certains utilisateurs comme la fermeture d'un lieu. Les machines finissent par s'arrêter, mais leur influence demeure visible dans l'histoire d'Emacs, de Lisp, des jeux d'aventure et du courrier électronique. Elles contribuent surtout à imposer une idée appelée à devenir centrale : un ordinateur relié à un réseau n'est pas seulement un instrument de calcul, mais aussi un espace dans lequel des personnes partagent des programmes, des informations et des connaissances.

9.3 Une famille brillante mais isolée

Le succès intellectuel des machines 36 bits ne garantit pas leur avenir industriel. Leur architecture, leurs systèmes et leur clientèle forment un monde spécifique. Les coûts de développement sont élevés, tandis que la croissance la plus rapide de **DEC** se produit autour des familles plus petites. Le DECsystem-10 puis le DECSYSTEM-20 prolongent la gamme, mais celle-ci finit par entrer en concurrence pour les ressources avec VAX.

Lorsque **DEC** décide d'abandonner la lignée, la réaction des utilisateurs montre la profondeur de leur attachement. Les logiciels et les habitudes ne se déplacent pas aussi facilement qu'un équipement. Une communauté de temps partagé constitue un patrimoine collectif : programmes, fichiers, pratiques et relations.

Cette leçon pèsera sur les choix futurs de l'entreprise. Il ne suffit plus de créer une meilleure machine. Il faut préserver l'investissement logiciel et offrir une trajectoire de migration. Le problème se pose bientôt à une échelle plus vaste avec le PDP-11, famille dont la réussite dépassera toutes les attentes et dont les limites deviendront, à leur tour, difficiles à abandonner.

Chapitre 10

Le PDP-11

À la fin des années 1960, **DEC** possède plusieurs familles répondant à des marchés différents. Les machines de 12 bits sont économiques, les modèles de 18 bits équipent notamment les laboratoires, tandis que les systèmes de 36 bits sont destinés au calcul scientifique et au temps partagé. Cette diversité a cependant un coût : chaque architecture nécessite ses propres logiciels et mobilise des équipes de développement distinctes.

DEC a déjà étudié une nouvelle famille, désignée sous le nom de PDP-X, mais le projet est abandonné. *Edson de Castro*, qui y a participé, quitte alors l'entreprise et fonde **Data General** avec plusieurs anciens ingénieurs de **DEC**. Leur Nova, présenté à la fin de 1968 puis livré à partir de 1969, est un mini-ordinateur de 16 bits compact et relativement peu coûteux. Son apparition montre qu'une jeune société peut concurrencer directement **DEC** sur son propre marché.

La réponse ne peut donc pas se limiter à une nouvelle amélioration du PDP-8. **DEC** doit concevoir une architecture de 16 bits capable de servir de base à plusieurs modèles, depuis les petites configurations économiques jusqu'aux systèmes plus puissants. L'objectif est de créer une véritable famille, et non une machine supplémentaire isolée des autres.

Les premières réflexions architecturales sont menées par *Gordon Bell* et *Harold McFarland* à l'université Carnegie-Mellon. *Harold McFarland* et *Roger Cady* établissent ensuite les spécifications détaillées de la nouvelle machine, tandis que d'autres ingénieurs participent à sa réalisation. Le PDP-11 naît ainsi d'un travail collectif et de compromis entre le coût du matériel, les performances et la facilité de programmation. Son architecture suffisamment régulière et extensible pourra être réalisée sous la forme de nombreux modèles sans perdre son identité.

10.1 Une architecture plus régulière

Annoncé en 1970, le PDP-11 travaille principalement avec des mots de 16 bits, mais chaque octet de sa mémoire possède sa propre adresse. Un mot occupe donc deux adresses consécutives. Deux caractères peuvent, par exemple, être placés aux adresses 1000 et 1001, et le programme peut lire ou modifier chacun d'eux séparément.

Sur une machine dont la mémoire est seulement adressée par mots, plusieurs caractères peuvent être regroupés dans un même mot. Pour en modifier un seul, le programme doit alors lire le mot entier, isoler le caractère recherché, puis replacer l'ensemble en mémoire. Le PDP-11 évite ces opérations grâce à ses instructions portant directement sur les octets. Le traitement des textes et des chaînes de caractères devient ainsi plus simple et plus naturel.

La machine possède huit registres de 16 bits. Six d'entre eux peuvent être employés aussi bien pour conserver une valeur que pour former une adresse. Les deux derniers ont des fonctions particulières : le registre R6 sert généralement de pointeur de pile, tandis que R7 contient l'adresse de la prochaine instruction à exécuter.

L'une des principales originalités du PDP-11 réside dans ses modes d'adressage. Dans les instructions comportant une source et une destination, chaque opérande est décrit de manière semblable par un registre et un mode d'adressage. Une instruction peut ainsi travailler sur des valeurs placées dans des registres, en mémoire ou à des adresses calculées au moment de l'exécution.

L'instruction suivante transfère un mot de 16 bits depuis l'adresse contenue dans R1 vers celle contenue dans R2 :

MOV (R1)+, (R2)+

Après le transfert, R1 et R2 sont automatiquement augmentés de deux. R1 désigne alors le mot source suivant et R2 l'emplacement de destination suivant. Une seule instruction réalise donc à la fois le transfert et l'avancement des deux pointeurs. Répétée dans une boucle, elle permet notamment de parcourir ou de copier un tableau de mots.

Pour copier une chaîne de caractères octet par octet, une boucle peut employer l'instruction correspondante :

MOVB (R1)+, (R2)+

L'instruction **MOVB** ne transfère qu'un octet. À chaque exécution, R1 et R2 sont donc augmentés d'une seule unité et désignent le caractère suivant dans chacune des deux zones de mémoire.

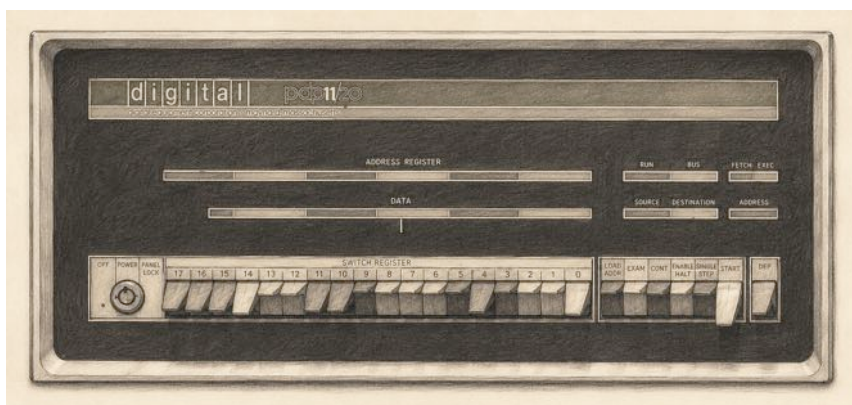


Figure 10.1 – Le panneau de commande du PDP-11/20.

Source : Illustration originale au crayon. D'après une photographie du panneau de commande d'un PDP-11/20.

Gordon Bell et ses coauteurs présentent cette architecture dans un article publié en 1970. Ils ne décrivent pas une machine idéale conçue sans contraintes, mais un compromis entre le coût du matériel, les performances et le coût de la programmation. La régularité du jeu d'instructions doit notamment faciliter le développement de logiciels plus importants sur un mini-ordinateur.

Cette organisation joue ensuite un rôle dans l'histoire du langage C. Lorsque les chercheurs des Bell Laboratories installent Unix sur un PDP-11, le langage B montre ses limites. Son modèle, hérité de machines organisées autour de mots entiers, convient mal à une architecture qui adresse directement les octets. Dennis Ritchie lui ajoute notamment des types permettant de distinguer les caractères, les entiers et les pointeurs. Cette évolution conduit progressivement au langage C, puis à la réécriture d'Unix dans ce langage en 1973.

Le PDP-11 n'est donc pas seulement une machine plus facile à programmer en assembleur. Son architecture fournit aussi le cadre matériel dans lequel se développent deux éléments essentiels de l'informatique moderne : le langage C et le système Unix.

10.2 L'UNIBUS

L'élément le plus visible de cette cohérence est l'UNIBUS.¹ Dans le PDP-11/20, le processeur, la mémoire et les contrôleurs de périphériques sont reliés à cette voie de communication commune. Les données peuvent y circuler dans les deux sens, d'un composant vers un autre.



Figure 10.2 – Un fond de panier UNIBUS de PDP-11 avec plusieurs cartes insérées. Les connecteurs distribuent les lignes du bus commun par lesquelles communiquent le processeur, la mémoire et les contrôleurs de périphériques.

Source : Photographie de Chrihern. Wikimedia Commons, domaine public.

L'UNIBUS fonctionne de manière asynchrone : les échanges ne sont pas réglés par une cadence identique pour tous les appareils. Lorsqu'un composant demande une opération, le destinataire répond lorsqu'il est prêt. Une mémoire rapide et un périphérique beaucoup plus lent peuvent ainsi employer le même bus sans devoir fonctionner à la même vitesse.

1. Un *bus* est un ensemble de lignes de communication partagées par plusieurs composants. Il transporte les adresses, les données et les signaux de contrôle.

Les périphériques s'intègrent également dans le même espace d'adressage que la mémoire. Leurs registres de commande et d'état reçoivent des adresses que le processeur peut lire ou modifier avec les instructions ordinaires. Pour lancer une opération sur un disque, par exemple, le programme écrit dans les registres de son contrôleur comme il écrirait dans une case mémoire. Le PDP-11 n'a donc pas besoin d'instructions d'entrée-sortie entièrement séparées.

Un contrôleur peut également demander la maîtrise du bus afin de transférer directement des données vers la mémoire ou depuis celle-ci. Le processeur n'a alors pas à prendre en charge chaque mot du transfert. Lorsque l'opération est terminée, le contrôleur peut l'en avvertir par une interruption. Ce mécanisme se révèle particulièrement utile pour les disques, les bandes magnétiques et les dispositifs d'acquisition.

Cette organisation facilite l'extension des machines. Un même type de contrôleur peut être employé sur plusieurs modèles compatibles, tandis que des fabricants tiers peuvent proposer leurs propres interfaces de communication, cartes d'acquisition ou systèmes de stockage. L'UNIBUS contribue ainsi à transformer le PDP-11 en une plate-forme autour de laquelle se développe tout un marché de périphériques.

Cette ouverture possède cependant ses limites. Tous les échanges se partagent la capacité du bus, qui finit par devenir un obstacle lorsque les processeurs, les mémoires et les disques gagnent en vitesse. Les modèles les plus puissants recevront donc des chemins spécialisés entre le processeur et la mémoire ou pour les périphériques rapides. Au début des années 1970, l'UNIBUS offre néanmoins un équilibre remarquable entre simplicité, coût et possibilités d'extension.

10.3 Le succès et les limites du PDP-11

Le PDP-11/20 est rapidement suivi de nombreux modèles. Des petites machines de table côtoient des systèmes installés dans plusieurs armoires, tandis que le LSI-11 rassemble le processeur sur une carte destinée aux configurations compactes et aux systèmes intégrés. Les performances, les capacités de mémoire et les périphé-

riques évoluent, mais l'architecture de base demeure reconnaissable.

Les systèmes d'exploitation se diversifient eux aussi. RT-11 convient aux petites configurations et aux applications en temps réel. RSX-11 répond aux besoins de systèmes plus complexes, notamment dans l'industrie, tandis que RSTS/E permet à plusieurs utilisateurs de travailler en temps partagé. À ces logiciels de **DEC** s'ajoutent UNIX et de nombreux programmes conçus par des universités, des laboratoires ou des entreprises indépendantes.

Cette continuité protège une partie des investissements du client. Celui-ci peut adopter un processeur plus puissant tout en conservant une grande partie de ses programmes, de ses données et de ses périphériques. La compatibilité n'est pas toujours parfaite, car les modèles ne possèdent pas tous les mêmes options, mais le passage d'une machine à une autre reste beaucoup plus simple qu'entre deux architectures sans rapport.

Le PDP-11 devient ainsi plus qu'une succession de processeurs : il forme une plate-forme. Un client accepte d'y investir parce qu'il pense pouvoir la faire évoluer sans repartir entièrement de zéro. Le principe rappelle celui du System/360 d'IBM, mais il est ici adapté au monde plus compact et plus accessible des mini-ordinateurs.

Cette réussite révèle cependant une faiblesse inscrite dans la conception d'origine. Une adresse de 16 bits ne permet de désigner que 65,536 octets, soit 64 Kio. Des mécanismes de gestion de la mémoire permettent aux systèmes ultérieurs d'utiliser une mémoire physique plus grande et de déplacer les programmes entre différentes zones. Un programme ne peut cependant pas accéder directement à toute cette mémoire dans un espace d'adressage unique. Certains modèles séparent également l'espace réservé aux instructions de celui qui contient les données, mais cette solution ne fait que repousser la limite.

À mesure que les logiciels grandissent, cette contrainte devient de plus en plus difficile à contourner. À partir de 1975, **DEC** prépare donc une extension de l'architecture dotée d'adresses de 32 bits : le VAX, dont le nom signifie *Virtual Address eXtension*. La nouvelle famille doit offrir un espace beaucoup plus vaste tout en préservant autant que possible les programmes, les périphériques, les données et les habitudes acquis autour du PDP-11.

Avant cette transition, la machine connaît toutefois un destin que **DEC** n'avait pas prévu. Aux Bell Laboratories, UNIX, d'abord développé sur un PDP-7, est porté sur le PDP-11 et y prend sa forme moderne. C'est également sur cette machine que le langage B évolue progressivement pour donner naissance au langage C. Le système et le langage finiront par dépasser largement l'ordinateur qui les a vus mûrir.

Chapitre 11

UNIX, C sur PDP-11

L'un des épisodes les plus durables de l'histoire du PDP-11 ne se déroule pas à Maynard, mais aux **Bell Laboratories**. Après le retrait des laboratoires du projet Multics, en 1969, un petit groupe de chercheurs cherche à préserver ce que cette expérience lui avait apporté : un environnement interactif dans lequel il est possible de créer, modifier et exécuter directement des programmes. *Ken Thompson* et *Dennis Ritchie* ont déjà commencé à construire Unix sur un PDP-7, mais cette machine vieillissante, qui ne leur appartient même pas, ne peut offrir une base durable à leur travail.

Au début de 1970, le groupe demande donc l'acquisition d'un PDP-11 nouvellement annoncé. Les propositions précédentes, trop coûteuses ou trop vagues, avaient été refusées. Cette fois, la demande s'appuie sur un objectif précis : réaliser un système destiné à l'édition et à la mise en forme de textes. L'idée est notamment portée par *Joseph Ossanna*, tandis que *Lee McMahon* contribue à obtenir le financement de la machine. La commande est finalement passée en mai 1970.

Cette histoire rappelle les débuts de **DEC** par un trait essentiel : un projet ambitieux progresse d'abord sous la forme d'une réalisation plus modeste et immédiatement utile. Les fondateurs de **DEC** avaient commencé par vendre des modules logiques avant de construire leur premier ordinateur. Aux Bell Laboratories, la préparation de documents permet de justifier l'achat de la machine sur laquelle Unix va poursuivre son développement. Dans les deux cas, le détour pratique ouvre une voie dont personne ne mesure encore toute la portée.

11.1 Après Multics

Les Bell Laboratories ont participé avec le MIT et **General Electric** au développement de Multics, système de temps partagé très ambitieux. Le projet veut servir de nombreux utilisateurs, offrir une protection élaborée et rendre les ressources informatiques disponibles comme un service. Sa complexité, ses délais et ses coûts conduisent toutefois les Bell Labs à se retirer en 1969.

Ken Thompson ne renonce pas à l'idée d'un environnement interactif. Il souhaite un système plus petit, qu'une équipe réduite puisse comprendre et modifier. Un PDP-7 peu utilisé lui fournit le terrain nécessaire. Avec *Dennis Ritchie* et quelques collègues, il construit progressivement un système de fichiers, des processus, un interpréteur de commandes et des outils simples.

La faiblesse de la machine impose l'économie. Il n'est pas possible de reproduire Multics en miniature. Les fonctions doivent être séparées, les programmes rester courts et les fichiers servir d'interface commune. Cette contrainte contribue à une philosophie qui deviendra célèbre : accomplir une tâche précise avec un outil compréhensible, puis combiner les outils plutôt que bâtir pour chaque besoin un programme gigantesque.

Le nom UNIX, d'abord écrit *Unics*, joue avec celui de Multics. Derrière le trait d'esprit se trouve un changement d'échelle. Le système n'est pas défini par un grand plan institutionnel, mais par les usages quotidiens de ceux qui l'écrivent. Ils emploient UNIX pour développer UNIX. L'éditeur sert à modifier le compilateur, le compilateur produit les commandes, et le système de fichiers organise les sources comme les documents.

Le PDP-7 devient rapidement trop limité. Le groupe souhaite un PDP-11, machine alors récente dont l'architecture traite naturellement les octets et offre des registres plus réguliers. Pour justifier l'achat, l'équipe propose de fournir un environnement de préparation de textes au service des brevets. La demande est acceptée. Le processeur arrive en 1970, avant certains périphériques nécessaires, et les premiers outils doivent encore être préparés à partir de l'ancienne machine.

11.2 Le passage au PDP-11

Lorsque le stockage devient disponible, UNIX migre réellement vers le PDP-11. Le nouveau processeur n'est pas seulement plus rapide. Il correspond mieux au type de logiciel que l'équipe veut écrire. Les octets adressables facilitent les textes et les fichiers, tandis que les modes d'adressage conviennent aux piles, aux tableaux et aux pointeurs. La machine reste assez petite pour qu'un groupe restreint en comprenne l'ensemble.

En 1971, le système assure un travail régulier de traitement documentaire. Cette utilisation apporte au projet une légitimité interne. UNIX n'est plus un exercice mené sur une machine délaissée : il rend un service et permet à ses auteurs d'obtenir du matériel plus puissant. La même année, la première édition du manuel des programmeurs commence à fixer les commandes et les conventions.

Le PDP-11 favorise une relation étroite entre matériel et logiciel. Lorsqu'une fonction manque, les chercheurs peuvent examiner les registres, les interruptions et les pilotes. Ils ne dépendent pas d'une couche opaque fournie par un constructeur lointain. La documentation de **DEC** et la régularité de l'architecture rendent cette exploration possible.

Cette proximité ne signifie pas qu'UNIX soit un produit de **DEC**. Les Bell Laboratories définissent leurs propres priorités et n'ont pas pour mission de promouvoir le PDP-11. Le constructeur fournit pourtant un environnement matériel particulièrement propice à cette expérimentation. En retour, UNIX donne à la machine une réputation académique et technique qui dépasse largement la publicité officielle.

Les petits programmes du système peuvent être reliés par des fichiers puis, bientôt, par des tubes de communication. Le résultat d'une commande devient l'entrée d'une autre. Cette composition fait du terminal un atelier logiciel : le programmeur construit une solution en assemblant des outils, comme les ingénieurs de Maynard avaient assemblé les premiers modules logiques.

Les travaux de Thompson et Ritchie montrent ainsi qu'une petite machine interactive peut accueillir un environnement complet, multiutilisateur et productif. Le PDP-11 n'est plus seulement

un contrôleur ou un ordinateur de laboratoire. Il devient le support d'une manière de programmer appelée à se diffuser dans les universités.

11.3 Un langage pour le système

Les premières versions d'UNIX sont largement écrites en assembleur. Cette situation donne un code efficace, mais attache le système à l'architecture. Thompson avait déjà conçu le langage B à partir de BCPL. Ritchie le fait évoluer pour mieux représenter les données et exploiter le PDP-11. Le nouveau langage reçoit le nom de C.

Dennis Ritchie n'essaie pas de masquer le matériel. C permet de manipuler des adresses, des bits et la disposition des données en mémoire. Il introduit cependant des types, des structures de contrôle et des fonctions qui donnent au programme une organisation plus claire que l'assembleur. Il se place dans une zone intermédiaire : assez proche de la machine pour écrire un noyau, assez abstrait pour qu'une grande partie du code survive à un changement de processeur.

Le PDP-11 influence ce développement. Ses octets, ses registres et ses modes d'adressage donnent une forme efficace à certaines expressions du langage. C ne constitue toutefois pas une transcription du jeu d'instructions. Son intérêt apparaît précisément lorsqu'il permet de décrire les structures du système sans répéter tous les détails du processeur.

En 1973, le noyau d'UNIX est réécrit en grande partie en C. Le système conserve nécessairement des portions dépendantes du matériel, notamment pour le démarrage et les entrées-sorties, mais la majorité du code peut être comprise et adaptée sans reprendre chaque instruction. Les portages réalisés ensuite démontrent qu'un système d'exploitation peut franchir la frontière d'une architecture avec un effort raisonnable. Cette évolution est documentée par Ritchie et par l'article que Thompson et lui publient en 1974.¹

1. Dennis M. RITCHIE. *The Evolution of the Unix Time-Sharing System*. 1984 ; Dennis M. RITCHIE. *The Development of the C Language*. 1993 ; Dennis M. RITCHIE et Ken THOMPSON. « The UNIX Time-Sharing System ». In :

La portabilité change progressivement les rapports de force. Dans le modèle traditionnel, le système d'exploitation accompagne une gamme de machines et disparaît avec elle. Avec UNIX et C, le logiciel peut devenir une continuité plus durable que le processeur. Les compétences du programmeur se déplacent avec le langage, les outils et les conventions.

Cette possibilité ne produit pas immédiatement un marché ouvert. Les portages restent complexes, les licences sont contrôlées et les matériels diffèrent profondément. Mais l'idée est installée : le code source peut porter une culture au-delà d'une machine particulière. Pour un constructeur intégré comme **DEC**, cette idée sera à la fois un avantage et une menace.

11.4 Expliquer et transmettre

Brian Kernighan occupe dans cette histoire une place distincte. Il ne crée pas le langage C et n'est pas l'auteur principal du noyau. Il participe cependant à l'environnement des Bell Laboratories, contribue à nommer et présenter plusieurs outils, et possède un talent particulier pour l'explication.

Avec Ritchie, il publie en 1978 *The C Programming Language*. Le livre est bref, dense et construit autour de programmes concrets. Il ne cherche pas à remplacer la pratique par une théorie exhaustive. Il donne au lecteur assez de règles et d'exemples pour commencer à écrire, puis l'amène vers les pointeurs, les structures et les entrées-sorties.

Cette clarté joue un rôle considérable. Un langage ne se diffuse pas seulement parce que son compilateur existe. Il faut des mots communs, un style et une manière de transmettre les bons usages. L'ouvrage de Kernighan et Ritchie fournit cette forme. Des générations de programmeurs apprennent C à travers des exemples dont l'arrière-plan matériel remonte au PDP-11, même lorsqu'ils travaillent sur une autre architecture.

La combinaison d'UNIX et de C donne également aux universités

un objet pédagogique exceptionnel. Le système est assez compact pour être étudié, son code source circule sous licence et le langage permet de relier les concepts de haut niveau aux mécanismes de la machine. Les étudiants peuvent lire un noyau réel au lieu d'une maquette construite uniquement pour l'enseignement.

Pour **DEC**, cette culture extérieure ajoute de la valeur au PDP-11. Les universités achètent ou conservent la machine parce qu'elle exécute UNIX, des chercheurs développent des logiciels qui circulent entre établissements, et les diplômés emportent leur expérience vers l'industrie. Le constructeur bénéficie donc d'un écosystème qu'il ne dirige pas.

Le paradoxe devient visible à long terme. UNIX et C donnent au PDP-11 ses lettres de noblesse logicielles, mais ils apprennent aussi aux utilisateurs à séparer le système de la machine. Lorsque les stations de travail et les microprocesseurs se multiplieront, le même environnement pourra passer d'un constructeur à l'autre. L'une des plus belles réussites associées à une machine **DEC** prépare ainsi un monde dans lequel le matériel propriétaire ne suffit plus à retenir le client.

Avant que cette conséquence n'apparaisse, le PDP-11 porte **DEC** vers une autre dimension. Les systèmes d'exploitation, les langages, les périphériques, la documentation et le service deviennent aussi importants que le processeur. La société cesse d'être seulement un fabricant de machines ; elle devient un fournisseur de mondes informatiques.

Chapitre 12

De la machine au système

Une architecture réussie attire des utilisateurs. Une entreprise durable doit ensuite les accompagner pendant des années. À mesure que le PDP-8, le PDP-10 et le PDP-11 s'installent dans les laboratoires et les entreprises, **DEC** découvre que le produit réel dépasse largement le processeur. Il comprend les périphériques, les systèmes d'exploitation, la documentation, la formation, les pièces de rechange et les personnes capables d'intervenir lorsqu'une machine s'arrête.

Cette extension transforme l'organisation. La société de modules, qui pouvait discuter directement avec chacun de ses premiers clients, devient un constructeur mondial. Le défi consiste à conserver la proximité technique tout en créant des procédures assez solides pour soutenir des milliers d'installations.

La diversité des systèmes ne relève pas d'une incapacité à choisir. Elle répond à des contraintes réelles. Un laboratoire temps réel veut une réponse déterministe et une petite empreinte mémoire. Une université privilégie le partage entre étudiants. Une entreprise demande fichiers, langages commerciaux et procédures d'exploitation. Au début, aucun système unique ne peut satisfaire ces priorités sans devenir trop lourd pour les petites configurations.

RT-11, RSX-11, RSTS/E et les autres environnements incarnent donc des compromis. Ils partagent une architecture matérielle, mais proposent des expériences différentes. Cette pluralité attire des marchés variés et donne aux équipes internes une grande autonomie. Elle crée aussi une dette : documentation, corrections et évolutions doivent être assurées pour plusieurs mondes en parallèle.

12.1 Une pluralité de logiciels

DEC ne cherche pas immédiatement à imposer un système d'exploitation unique. Les petites configurations du PDP-8 utilisent des moniteurs, puis OS/8. Le PDP-11 accueille RT-11, RSX-11, RSTS/E et d'autres environnements. Les machines 36 bits disposent de TOPS-10 et TOPS-20. Cette pluralité répond à des besoins différents : contrôle en temps réel, enseignement, calcul interactif ou traitement d'entreprise.

Elle donne aux clients une grande liberté, mais augmente le coût de maintenance. Chaque système possède ses outils, ses pilotes et ses habitudes. Les langages doivent être adaptés à plusieurs architectures. Les corrections circulent entre équipes. Lorsque le matériel change, le logiciel devient un capital qu'il faut protéger.

Les manuels jouent ici un rôle stratégique. Les *handbooks* décrivent les registres, les interfaces et les exemples de programmation. Les guides de maintenance permettent de suivre un signal jusqu'à la carte défailante. La documentation n'est pas seulement un accompagnement commercial : elle autorise l'utilisateur à comprendre et parfois à modifier son système.

Le logiciel devient progressivement une condition de la vente. Un processeur plus rapide ne suffit pas si le client doit récrire son application ou ne trouve pas le langage dont il a besoin. **DEC** investit dans les compilateurs, les éditeurs, les bibliothèques et les diagnostics. Les catalogues s'épaississent. La société ne vend plus seulement une architecture : elle promet un environnement de travail.

Cette promesse rapproche les équipes de logiciel des ingénieurs de terrain. Une fonction qui paraît secondaire au concepteur peut être décisive pour une installation. Les retours des clients alimentent les versions suivantes, mais ils rendent également le système plus complexe. Chaque demande satisfaite devient une compatibilité à préserver.

12.2 La force de DECUS

Le **Digital Equipment Computer Users Society**, créé au début des années 1960, organise l'échange de programmes et d'expérience entre utilisateurs. Des bibliothèques de logiciels circulent sur bandes et rubans. Les réunions rassemblent chercheurs, programmeurs, responsables de sites et ingénieurs de **DEC**. Un participant peut arriver avec une question précise et repartir avec un utilitaire écrit dans une autre université.

Un badge resté célèbre invite à « voler à ses amis ». La formule ne célèbre pas le pillage, mais la circulation des bonnes solutions dans un monde où chaque site ne peut pas tout développer. DECUS augmente la valeur des machines sans que le constructeur écrive lui-même tous les programmes. Il offre également à **DEC** un observatoire privilégié des besoins réels.¹

Cette proximité peut être exigeante. Les utilisateurs techniques comparent les performances, signalent les défauts et demandent des améliorations. Mais elle crée une fidélité rare. Acheter une machine **DEC**, c'est entrer dans une communauté de pratiques, pas seulement signer un contrat de matériel.

Les réunions DECUS donnent à cette relation un visage. Les utilisateurs présentent leurs programmes, échangent des rubans ou des listings et critiquent directement les choix du constructeur. Les discussions ne sont pas toujours confortables pour **DEC**, mais elles fournissent une information qu'une enquête commerciale classique capterait mal. Les participants connaissent les détails du système et décrivent les conséquences concrètes d'une décision.

La bibliothèque de programmes réduit la duplication. Un laboratoire peut adapter une routine écrite ailleurs au lieu de recommencer. Cette circulation rappelle le MIT, mais elle est désormais organisée à l'échelle d'une clientèle internationale. DECUS devient une institution relativement autonome, suffisamment proche du constructeur pour influencer ses produits et suffisamment indépendante pour défendre les utilisateurs.

1. COMPUTER HISTORY MUSEUM. *"Steal from Your Friends" DECUS Button*. 1965.

12.3 Le service comme promesse

Une installation industrielle ne peut attendre plusieurs semaines une pièce ou un technicien. **DEC** développe donc un réseau de service de terrain. Les ingénieurs disposent de diagnostics, de pièces de rechange et d'une documentation détaillée. Ils doivent comprendre le système dans son contexte : un arrêt peut interrompre une expérience, une production ou un service administratif.

La relation avec le client se prolonge sur toute la durée de vie de la machine. Les mises à jour de logiciel, l'ajout d'un disque ou le remplacement d'un processeur deviennent des occasions de vente, mais aussi des responsabilités. La qualité du service renforce la réputation de **DEC** et justifie le choix d'une plate-forme cohérente.

Les fabricants OEM complètent cet ensemble. Ils intègrent les processeurs et les bus à des produits spécialisés, tandis que des sociétés tierces proposent mémoires, contrôleurs et interfaces. L'ouverture crée un marché autour de **DEC**, parfois au prix d'une concurrence avec ses propres périphériques.

Le service de terrain est le lieu où l'intégration devient réelle. Le technicien ne peut pas déclarer que le processeur fonctionne si le client ne parvient pas à lire son disque ou à démarrer son application. Il doit diagnostiquer l'ensemble, conserver des pièces et parfois improviser une solution avant l'arrivée d'une correction officielle.

Cette présence coûte cher, mais elle construit la confiance. Les systèmes contrôlent des expériences, des lignes industrielles ou des dossiers administratifs ; leur arrêt a des conséquences immédiates. Le contrat de maintenance transforme donc la vente initiale en relation durable. Il donne aussi à **DEC** un accès constant aux problèmes de ses clients.

L'entreprise développe des diagnostics qui testent les cartes, les mémoires et les périphériques. Ces programmes, souvent moins visibles que les applications, participent à la qualité de la plate-forme. Ils prolongent l'obsession de Ken Olsen pour les systèmes observables et réparables.

12.4 Changer d'échelle sans perdre son identité

À la fin des années 1970, **DEC** n'est plus une petite société de Maynard. Elle possède des usines, des filiales, des centres de formation et une présence internationale. Des rédacteurs, formateurs, techniciens et commerciaux rejoignent les architectes et les programmeurs. **Digital Press** publie des ouvrages qui dépassent la documentation immédiate des produits, notamment *Computer Engineering*, où *Gordon Bell*, *Craig Mudge* et *John McNamara* analysent les machines comme les étapes d'une évolution.

Cette croissance renforce la capacité de l'entreprise à fournir des systèmes complets. Elle rend aussi chaque rupture plus coûteuse. Un nouveau processeur doit préserver les logiciels, les périphériques et les compétences des clients. Le PDP-11 a construit une base immense, mais son espace d'adressage devient insuffisant. Abandonner cette base serait dangereux ; la prolonger sans limite paraît impossible.

La solution prendra le nom de VAX. Elle devra accomplir un exercice que **DEC** n'avait encore jamais mené à cette échelle : changer profondément l'architecture tout en donnant au client le sentiment d'une continuité.

Quatrième partie

L'empire VAX

Chapitre 13

Évoluer sans rompre

Au milieu des années 1970, le PDP-11 est à la fois la plus grande réussite de **DEC** et l'un de ses principaux problèmes. Les utilisateurs ont investi dans des programmes, des périphériques et des équipes formées. Pourtant, l'espace d'adressage de 16 bits limite les applications les plus ambitieuses. Les extensions ajoutées aux modèles supérieurs compliquent la programmation sans supprimer la contrainte fondamentale.

Une rupture complète ferait perdre l'avantage de la base installée. Une simple amélioration ne suffirait pas. Le comité réuni en 1975 doit donc concevoir une architecture nouvelle qui paraisse familière aux utilisateurs du PDP-11. **DEC** emploiera l'expression « compatibilité culturelle » : mêmes habitudes générales, mêmes types de données, outils proches et chemin de migration crédible, même lorsque le matériel change profondément.

Les extensions du PDP-11 montrent les limites d'une croissance par ajouts. Des espaces séparés pour les instructions et les données, des mécanismes de gestion mémoire et des modèles supérieurs permettent de dépasser certaines barrières, mais le programmeur doit connaître des règles de plus en plus complexes. L'élégance initiale risque de disparaître sous les solutions de transition.

Le comité VAX examine donc le problème comme une continuité culturelle plutôt que comme une compatibilité instruction par instruction. Le client doit reconnaître les tailles de données, les conventions, les outils et la manière d'administrer le système. Cette notion est moins rigide qu'une reproduction exacte du PDP-11, mais plus exigeante qu'une simple passerelle de fichiers. Elle oblige les équipes à penser la migration dès la conception.

13.1 Étendre l'adresse

William Strecker joue un rôle majeur dans la définition de la nouvelle architecture. Le nom VAX signifie *Virtual Address eXtension*. Le passage à des adresses de 32 bits ouvre un espace théorique de quatre milliards d'octets, sans exiger que toute cette mémoire soit physiquement installée. Le système d'exploitation peut donner à chaque processus un espace d'adresses cohérent et déplacer les pages entre mémoire et disque selon les besoins.¹

L'architecture augmente également le nombre de registres et généralise les opérations sur plusieurs tailles de données. Son jeu d'instructions est abondant. Cette complexité reflète la conviction de l'époque qu'une machine peut prendre en charge des opérations de haut niveau et simplifier le travail des compilateurs.

Le projet porte le nom de code *Star*. Il ne consiste pas à construire seulement un processeur. Matériel et système d'exploitation sont développés ensemble. Cette coordination représente un changement d'échelle pour **DEC** : le logiciel n'attendra plus la fin de la conception matérielle pour s'adapter.

La mémoire virtuelle ne sert pas seulement à exécuter de gros programmes. Elle isole les utilisateurs, donne au système un moyen de déplacer les pages et simplifie la représentation d'espaces de travail distincts. Pour un environnement multiutilisateur, cette protection est fondamentale. Une erreur ne doit pas permettre à un programme d'écraser le noyau ou les données d'un voisin.

Le jeu d'instructions reflète également les ambitions de l'époque. Certaines opérations complexes peuvent manipuler des chaînes ou faciliter les appels de procédures. Les concepteurs espèrent réduire l'écart entre les langages de haut niveau et le matériel. Cette richesse sera plus tard critiquée par les partisans des architectures RISC, mais elle correspond au contexte technologique et logiciel du milieu des années 1970.

1. La *mémoire virtuelle* sépare les adresses utilisées par un programme de leur emplacement physique. Elle facilite l'isolation des processus et permet d'exécuter des programmes dont l'espace logique dépasse la mémoire réellement présente.

13.2 Le VAX-11/780

Le premier modèle, le VAX-11/780, est présenté en 1977. Son nom conserve explicitement la filiation avec le PDP-11. La machine peut exécuter certains programmes PDP-11 dans un mode de compatibilité, mais l'objectif principal est de faciliter une migration progressive vers le nouveau monde 32 bits.

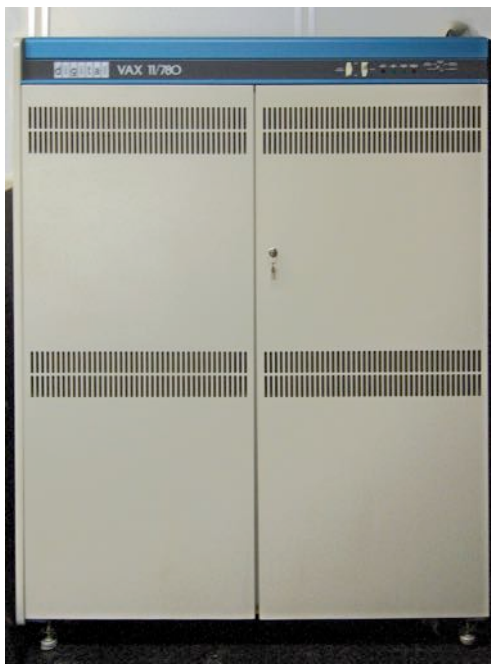


Figure 13.1 – DEC VAX-11/780

Source : Emiliano Russo, Associazione Culturale VerdeBinario.
Domaine public, Wikimedia Commons.

Le 11/780 occupe une position inhabituelle. Il offre des capacités comparées à celles de systèmes beaucoup plus coûteux, tout en restant dans la culture du mini-ordinateur : interaction, périphériques variés, logiciels scientifiques et possibilité d'installer le système au niveau d'un département. Le Computer History Museum souligne que le succès de la famille VAX transforme **DEC** en deuxième constructeur informatique mondial.²

2. William D. STRECKER. « VAX-11/780 : A Virtual Address Extension to the DEC PDP-11 Family ». In : *AFIPS Conference Proceedings* 47 (1978), p. 967-980 ; COMPUTER HISTORY MUSEUM. *VAX-11/780*. 1978.

La machine devient également une unité de comparaison informelle. Le terme VUP, pour *VAX Unit of Performance*, mesure les systèmes par rapport au 11/780. Cette habitude montre combien le modèle s'impose comme référence, même si elle simplifie des performances qui varient selon les applications.

Le lancement du 11/780 met en scène la continuité. Le nom, la documentation et le mode de compatibilité rassurent les utilisateurs du PDP-11, tandis que les démonstrations soulignent l'espace mémoire et la puissance. **DEC** ne demande pas aux clients d'oublier leur histoire ; elle leur propose de la prolonger.

Le succès transforme le centre de gravité de l'entreprise. Les petites machines restent importantes, mais VAX attire les applications dont dépendent des services entiers. Les contrats grossissent, les cycles de décision s'allongent et la comparaison avec IBM devient plus directe. **DEC** n'est plus seulement le constructeur qui contourne le grand système : il commence à occuper une partie de son territoire.

13.3 Une famille plutôt qu'un sommet

Le véritable pari n'est pas de vendre un seul grand modèle. **DEC** veut décliner VAX du petit système au calculateur puissant. Le VAX-11/750 puis le 11/730 réduisent le coût. Les MicroVAX installent l'architecture dans des boîtiers plus compacts. Les VAX 8000 et 6000 montent vers des configurations proches du grand système. Les VAXstations l'emmènent sur le bureau des ingénieurs.

Cette largeur de gamme répond à la stratégie de plate-forme qui avait fait la force du System/360, mais avec une différence importante : **DEC** cherche à relier des machines distribuées plutôt qu'à tout concentrer dans un centre unique. Un client peut placer des VAX dans plusieurs services, les connecter et conserver un environnement logiciel commun.

L'architecture apporte ainsi une continuité au matériel. Pour qu'elle devienne un empire, il faut encore un système d'exploitation capable de protéger les programmes, les données et les habitudes. Cette tâche revient à une équipe dirigée par un ingénieur réputé pour son exigence et son indépendance : *Dave Cutler*.

Chapitre 14

Le logiciel comme héritage

Dave Cutler rejoint **DEC** au début des années 1970 et travaille sur les systèmes temps réel du PDP-11. Il dirige notamment le développement de RSX-11M, environnement compact et rigoureux destiné à des applications qui ne peuvent accepter une réponse imprévisible. Cette expérience le prépare à une tâche beaucoup plus vaste : construire le système d'exploitation de VAX pendant que l'architecture matérielle elle-même prend forme.

Le nouveau système reçoit le nom de VMS, pour *Virtual Memory System*. Sa première version est livrée en 1978. VMS doit exploiter la mémoire virtuelle, servir plusieurs utilisateurs, protéger leurs travaux et offrir les langages attendus par les clients scientifiques comme commerciaux. Il doit surtout donner l'impression que VAX prolonge l'investissement réalisé autour du PDP-11 au lieu de l'annuler.

Cutler vient d'une culture où le système doit respecter des délais et où un défaut ne peut être masqué par une interface élégante. Son équipe travaille avec une forte identité, parfois en tension avec d'autres groupes de **DEC**. Cette indépendance favorise la cohérence de VMS. Elle nourrit aussi le style de management que Cutler emportera plus tard chez **Microsoft**.

Le développement simultané exige des simulateurs, des prototypes et des accords constants sur les interfaces. Le système doit être écrit avant que le matériel définitif existe en quantité. Les erreurs découvertes dans l'un peuvent obliger à modifier l'autre. Cette interdépendance augmente le risque, mais elle évite de livrer un processeur sans environnement exploitable.

14.1 Concevoir avec la machine

Dans de nombreux projets, le logiciel arrive après le matériel et doit s'accommoder de décisions déjà prises. Pour VAX, les équipes échangent en permanence. Les besoins du système influencent les mécanismes de protection, les interruptions et la gestion de mémoire. Les possibilités du processeur, en retour, déterminent l'organisation du noyau et des compilateurs.

Cette coopération n'élimine pas les tensions. Les architectes veulent une machine élégante et complète; les développeurs du système cherchent des mécanismes fiables, testables et livrables. Cutler est connu pour défendre avec vigueur les besoins de son équipe. Le projet réussit parce que le conflit reste orienté vers un même objectif : faire fonctionner un ensemble, pas seulement réussir une démonstration de processeur.

VMS fournit des processus isolés, un système de fichiers structuré, des mécanismes de sécurité, des files d'impression, des outils de développement et le langage de commandes DCL. Il accueille FORTRAN, BASIC, COBOL, Pascal, PL/I et d'autres langages. La plate-forme s'adresse ainsi aussi bien aux laboratoires qu'aux services de gestion.

DCL offre une interface de commandes structurée, avec des conventions stables pour lancer les programmes, manipuler les fichiers et administrer les travaux. Elle n'imité pas UNIX; elle reflète une conception plus intégrée du système. Les noms, les privilèges et les procédures sont pensés pour des installations où plusieurs services partagent une machine administrée professionnellement.

Le système de fichiers et les mécanismes de versions répondent au travail quotidien. Conserver plusieurs générations d'un fichier réduit le risque d'une modification malheureuse. Les files d'impression, les comptes et les journaux transforment le processeur en service collectif. La puissance de VAX devient visible à travers une organisation logicielle plutôt qu'à travers le panneau avant.

14.2 Migrer sans tout recommencer

La compatibilité ne se réduit pas à l'exécution de quelques instructions anciennes. **DEC** fournit des outils, des conventions et des guides de migration. Les clients peuvent transférer progressivement leurs données et réécrire les composants qui exigent les nouvelles capacités. Les programmeurs retrouvent des noms, des concepts et une documentation familière.

Cette continuité possède une valeur économique considérable. Un programme scientifique ou industriel représente parfois plusieurs années de travail. Sa disparition coûterait davantage que le processeur. En protégeant le logiciel, **DEC** rend acceptable l'achat d'une architecture nouvelle.

Le succès produit toutefois un attachement croissant. Plus VMS devient complet, plus il devient difficile de le remplacer. Chaque fonction supplémentaire augmente la valeur de la plate-forme, mais aussi le poids de sa maintenance. Le système n'est plus un accompagnement du matériel : il constitue un patrimoine autonome.

La migration est aussi humaine. Les opérateurs et programmeurs formés au PDP-11 ne doivent pas se sentir étrangers. Les manuels, les cours et les outils de conversion accompagnent le matériel. Le coût de cette transition montre que la compatibilité est un produit en soi : elle se conçoit, se documente et se maintient.

VMS renforce ainsi le lien entre client et constructeur. Plus l'installation utilise ses mécanismes de sécurité, de fichiers et de réseau, plus un changement de plate-forme devient coûteux. Cette fidélité assure des revenus durables, mais elle peut se transformer en dépendance mutuelle. **DEC** doit continuer à faire évoluer le système sans rompre les applications anciennes.

14.3 La disponibilité comme argument

Les clients confient aux VAX des activités qui ne doivent pas s'interrompre. VMS développe donc des mécanismes de journalisation, de sauvegarde, de contrôle des accès et de reprise. La stabilité devient un argument commercial aussi important que la vitesse.

Cette orientation prépare les VAXclusters, où plusieurs machines

pourront partager des ressources et continuer à servir les utilisateurs malgré certaines pannes. Le système d'exploitation doit alors coordonner des processeurs distincts, gérer les accès concurrents aux fichiers et présenter l'ensemble comme un environnement cohérent.

VMS devient plus tard OpenVMS, signe de l'effort d'ouverture et de l'évolution des standards. Mais son identité demeure liée à la conception intégrée de VAX. Lorsque *Dave Cutler* quitte **DEC** en 1988 pour rejoindre **Microsoft** et diriger le développement de Windows NT, il emporte une expérience des systèmes robustes et portables acquise à Maynard. L'influence de **DEC** se diffuse ainsi jusque dans un concurrent qui dominera le logiciel du PC.

La plate-forme VAX/VMS atteint sa pleine force lorsque les machines cessent d'être des îlots. Réseau, terminaux et stockage vont former un ensemble distribué que les clients perçoivent comme un seul système de travail.

Chapitre 15

Le réseau devient le système

Le mini-ordinateur avait rapproché le calcul d'un laboratoire ou d'un département. Le réseau permet désormais de rapprocher les machines entre elles. Pour **DEC**, cette évolution n'est pas un complément tardif. Dès le milieu des années 1970, l'entreprise conçoit une architecture de communication destinée à relier ses différentes familles et leurs systèmes d'exploitation.

DECnet est annoncé en 1975. Ses premières versions répondent surtout aux besoins des systèmes RSX, puis l'architecture s'étend à VMS, TOPS, RSTS et d'autres environnements. Un utilisateur peut copier un fichier, accéder à une ressource distante ou lancer un travail sur une autre machine. Le réseau ne sert plus seulement à connecter des terminaux à un processeur central : il relie des ordinateurs capables d'agir chacun de manière autonome.

La mise en réseau prolonge une idée ancienne de **DEC**. Le mini-ordinateur avait donné de l'autonomie à un laboratoire en lui évitant de dépendre entièrement du centre de calcul. Il serait contradictoire de reprendre cette autonomie au nom du partage. Le réseau doit donc relier des machines qui restent capables de travailler seules.

Cette conception diffère du simple branchement de terminaux sur un grand ordinateur. Chaque nœud possède son système, ses disques et ses utilisateurs. La communication ajoute des possibilités sans supprimer l'identité locale. Une équipe peut conserver son VAX ou son PDP-11 tout en échangeant avec les autres services.

Le défi est autant logiciel que matériel. Les systèmes d'exploitation de **DEC** n'ont pas été créés au même moment ni pour les mêmes usages. Leur faire partager des fichiers et des noms oblige à définir des protocoles communs. DECnet devient ainsi une architecture transversale qui tente de réunir la diversité accumulée par l'entreprise.

15.1 Une entreprise distribuée

Cette vision correspond à la structure des clients de **DEC**. Les laboratoires, les usines et les services possèdent souvent leurs propres machines. Les remplacer par un centre unique ferait perdre la proximité acquise. DECnet propose une autre solution : conserver le calcul local tout en partageant données et services.

La phase III, annoncée en 1980, permet de construire des réseaux de plus de deux cents nœuds et fonctionne sur plusieurs systèmes d'exploitation. La complexité est importante, mais l'utilisateur voit progressivement apparaître un espace commun. Les noms de machines et de fichiers distants entrent dans les habitudes quotidiennes.¹

La même année, **DEC**, **Intel** et **Xerox** coopèrent à la normalisation commerciale d'Ethernet. Le réseau local à paquets offre un moyen commun de connecter processeurs, serveurs de terminaux et périphériques. **DEC** doit désormais faire coexister ses protocoles avec une technologie destinée à devenir un standard multiconstructeur.

Pour l'administrateur, le réseau apporte de nouvelles responsabilités. Une panne n'est plus nécessairement située dans la machine devant laquelle se trouve l'utilisateur. Elle peut venir d'une ligne, d'un routeur, d'un nom mal configuré ou d'un service distant. **DEC** doit former des spécialistes capables de raisonner à l'échelle d'un ensemble distribué.

Pour l'utilisateur, en revanche, le succès se mesure à la disparition de cette complexité. Copier un fichier vers une autre machine ou ouvrir une session distante doit devenir une opération ordinaire. Le réseau cesse d'être un projet expérimental lorsqu'il entre dans la syntaxe quotidienne des commandes.

15.2 Le terminal devient une norme

Le VT100, lancé en 1978, montre comment un périphérique peut dépasser la gamme qui l'a produit. Le terminal respecte

1. DIGITAL EQUIPMENT CORPORATION. *Networks Timeline*. Digital Computing History Timeline. 1997.

les séquences de contrôle normalisées par l'ANSI et utilise un microprocesseur Intel 8080 pour gérer plusieurs jeux de caractères, l'affichage en gras et les fonctions de l'écran. Son succès pousse d'autres fabricants à reproduire son comportement.²

Pour l'utilisateur, le VT100 est le visage du système. Il affiche l'éditeur, le courrier, les commandes et les résultats, qu'ils proviennent d'un PDP-11, d'un VAX ou d'une machine distante. Le processeur peut changer derrière le réseau ; le clavier et l'écran maintiennent une continuité sensible.

Cette standardisation contient déjà une ambiguïté. Elle facilite la vente de systèmes **DEC**, mais rend aussi le terminal utilisable avec des ordinateurs concurrents. Une interface devenue norme échappe en partie à son créateur.

Le VT100 connaît un succès qui dépasse la simple vente de terminaux. Ses séquences de contrôle deviennent un langage partagé entre logiciels et écrans. Un éditeur peut déplacer le curseur, effacer une zone et mettre en évidence un texte sans connaître le détail du matériel. Des décennies plus tard, les émulateurs de terminaux continueront à reproduire ce comportement.

Cette postérité illustre une loi paradoxale. Lorsqu'un constructeur réussit à imposer une interface, celle-ci devient utile à ses concurrents. Le standard augmente le marché, mais réduit l'exclusivité. **DEC** gagne avec le VT100 une influence durable et, en même temps, contribue à un monde où le terminal peut être séparé de la machine qui l'a popularisé.

15.3 VAXcluster et le stockage commun

Annoncé en 1983, VAXcluster relie plusieurs VAX exécutant chacun VMS et leur permet de partager des fichiers et des périphériques de stockage. Un gestionnaire de verrous distribués coordonne les accès concurrents. Les contrôleurs HSC administrent des ensembles de disques et prennent en charge certaines fonctions sans mobiliser le processeur principal.

2. DIGITAL EQUIPMENT CORPORATION. *VT100 User Guide*. Digital Equipment Corporation. 1978 ; COMPUTER HISTORY MUSEUM. *VT100 Terminal*. 1978.

Le résultat ne correspond ni à un ordinateur unique ni à un simple réseau de postes indépendants. Plusieurs machines coopèrent et peuvent offrir une disponibilité supérieure. Lorsqu'un processeur est arrêté pour maintenance, une partie des services peut continuer ailleurs. Le Computer History Museum considère le HSC comme l'un des produits de stockage les plus réussis de **DEC**.³

Les clusters renforcent la fidélité à VMS et aux périphériques du constructeur. Ils donnent également une forme concrète à l'idée d'informatique distribuée : la puissance et les données ne résident plus nécessairement dans une seule armoire.

VAXcluster répond à une demande de continuité de service. Les clients ne veulent plus seulement une machine fiable ; ils veulent que l'application reste disponible pendant la maintenance ou malgré certaines défaillances. Le cluster déplace donc la notion de fiabilité du composant vers l'ensemble. Une machine peut s'arrêter sans que le service entier disparaisse.

Cette idée change l'organisation des salles informatiques. Les disques sont placés dans des ressources communes, plusieurs processeurs coopèrent et les administrateurs planifient les arrêts. La puissance peut être augmentée par étapes. Le client ne remplace plus nécessairement une machine par une plus grande : il ajoute un nœud à un environnement existant.

La réussite repose sur l'intégration de VMS, de DECnet et du stockage. Aucun de ces éléments, pris isolément, n'explique le résultat. Le cluster est l'expression la plus aboutie du modèle **DEC** : une architecture de système où matériel et logiciel ont été pensés ensemble.

15.4 Un monde cohérent, mais de moins en moins fermé

À son apogée, **DEC** peut fournir le processeur, le système, le terminal, les disques, le réseau et le service. Pour un client, cette cohérence réduit les risques. Un interlocuteur unique prend en

3. COMPUTER HISTORY MUSEUM. *Networked Storage Systems Commercialized*. 1983.

charge l'ensemble et garantit que les composants ont été conçus pour fonctionner ensemble.

Mais Ethernet, TCP/IP, UNIX et les terminaux standardisés modifient le marché. Les clients veulent relier des équipements de marques différentes. L'ouverture devient une exigence, pas seulement une option. **DEC** tente d'intégrer les standards à DECnet et de proposer des stratégies multiconstructeurs, mais la valeur se déplace progressivement de l'ensemble propriétaire vers les composants interchangeables.

Le changement devient évident lorsque le terminal passif cède la place au micro-ordinateur et à la station de travail. L'utilisateur ne veut plus seulement accéder à la puissance du réseau : il veut posséder de la puissance sur son bureau. C'est un déplacement que **DEC**, malgré toute son expérience des petites machines, aura du mal à transformer en avantage.

Chapitre 16

L'informatique personnelle échappe à DEC

L'histoire de **DEC** pourrait laisser croire que l'entreprise était naturellement préparée à l'arrivée de l'ordinateur personnel. Depuis la fin des années 1950, elle avait précisément contribué à rapprocher le calcul de l'utilisateur. Les PDP avaient quitté les grands centres informatiques pour entrer dans les laboratoires, les universités, les ateliers et les services techniques. Avec le VAX, cette logique s'était prolongée à une échelle plus importante.

À première vue, le micro-ordinateur semble donc poursuivre le même mouvement : rendre la machine plus petite, moins coûteuse et plus directement accessible.

Mais le changement qui s'amorce à la fin des années 1970 ne concerne pas seulement la taille des ordinateurs. Il transforme également la manière de les concevoir, de les vendre et surtout de les choisir.

Avec l'informatique personnelle, la machine cesse progressivement d'être un système complet défini par un seul constructeur. Le client accorde une importance croissante aux logiciels disponibles, aux périphériques compatibles, aux formats de fichiers et à la possibilité de remplacer facilement une machine par une autre.

Le centre de gravité de l'industrie commence à se déplacer.

16.1 Un marché qui change de règles

Au début des années 1980, l'ordinateur personnel devient un produit vendu en très grandes quantités. Il ne s'adresse plus seulement aux ingénieurs ou aux laboratoires. Il entre dans les entreprises, les administrations et progressivement chez les particuliers.

L'**IBM PC**, lancé en 1981, joue un rôle majeur dans cette trans-

formation. Sa conception repose sur des composants disponibles sur le marché et son système d'exploitation est fourni par **Microsoft**. Très rapidement, d'autres constructeurs produisent des machines capables d'exécuter les mêmes logiciels.

Un standard de fait se forme ainsi autour du PC compatible.

Cette situation est très différente du modèle qui avait fait le succès de **DEC**. Un client achetant un PDP ou un VAX entraînait généralement dans un ensemble cohérent comprenant le processeur, le système d'exploitation, les périphériques, les outils de développement et le service du constructeur.

Dans le monde du PC, ces éléments commencent au contraire à provenir de fournisseurs différents. Le processeur peut être conçu par une entreprise, le système d'exploitation par une autre et les applications par une multitude d'éditeurs indépendants.

La valeur d'une machine ne dépend donc plus uniquement de ses qualités techniques. Elle dépend de plus en plus de l'écosystème auquel elle donne accès.

Cette évolution modifie également les circuits commerciaux. Un mini-ordinateur ou un VAX est souvent vendu comme un système complet, accompagné d'une installation et d'un suivi technique. Le PC peut être acheté en quantité, installé rapidement et remplacé quelques années plus tard.

Les volumes augmentent, les prix diminuent et le renouvellement des machines s'accélère.

Pour une entreprise habituée à maintenir ses architectures pendant de longues périodes, ce nouveau rythme est difficile à adopter.

16.2 Trois réponses au lieu d'une

DEC ne reste pourtant pas absente du marché de l'informatique personnelle. Au début des années 1980, elle propose plusieurs machines destinées à un utilisateur individuel.

Le problème est précisément qu'elle en propose plusieurs.

Le DECmate prolonge l'univers 12 bits issu du PDP-8 et s'inscrit notamment dans la continuité des systèmes de traitement de texte de l'entreprise.

Le Professional 300 reprend pour sa part des éléments de l'architecture PDP-11 afin de les adapter à une machine individuelle.

Le Rainbow 100 suit une autre direction. Il cherche à rejoindre le monde des micro-ordinateurs déjà existants et à exécuter des logiciels provenant notamment des environnements CP/M et MS-DOS.

Ces trois choix correspondent à des logiques techniques compréhensibles. Chacun tente de préserver ou d'étendre une partie du patrimoine de **DEC**.

Mais vus du marché, ils donnent surtout l'image d'une stratégie dispersée.

Les machines ne partagent pas nécessairement les mêmes logiciels ni les mêmes périphériques. Un client intéressé par un ordinateur personnel **DEC** doit donc choisir non seulement une machine, mais également l'univers technique qui l'accompagne.

Cette situation devient particulièrement problématique au moment où le marché évolue dans la direction opposée. Les utilisateurs cherchent précisément à réduire leur dépendance envers une architecture particulière. Ils veulent pouvoir retrouver leurs logiciels sur différentes machines et choisir leurs périphériques parmi un grand nombre de fournisseurs.

Le succès du compatible IBM repose largement sur cette logique.

Le problème de **DEC** n'est donc pas un manque de compétence technique. L'entreprise dispose d'ingénieurs capables de concevoir d'excellentes machines.

La difficulté est ailleurs : choisir une plate-forme unique aurait probablement obligé l'entreprise à abandonner une partie de ses propres architectures et à accepter des standards définis en dehors d'elle.

Or la culture de **DEC** s'est construite autour d'une forte maîtrise de l'ensemble du système.

16.3 Le poids de l'organisation

La coexistence du DECmate, du Professional et du Rainbow n'est pas seulement une erreur de catalogue. Elle révèle également

la manière dont l'entreprise s'est développée.

Au cours des décennies précédentes, plusieurs groupes internes ont conçu leurs propres machines, leurs propres périphériques et parfois leurs propres solutions logicielles. Cette organisation avait favorisé l'expérimentation et permis à **DEC** de faire évoluer rapidement ses familles de mini-ordinateurs.

Dans un marché encore ouvert, cette diversité constitue une force.

Mais lorsque des standards extérieurs commencent à s'imposer, elle devient plus difficile à gérer.

Chaque groupe possède ses clients, ses compétences et ses intérêts. Abandonner une architecture signifie renoncer à des années de développement et parfois à une base installée importante.

L'entreprise se retrouve ainsi confrontée à un paradoxe. La diversité technique qui avait nourri son innovation rend plus difficile la concentration de ses efforts sur une seule réponse au marché du PC.

Pendant ce temps, les compatibles IBM renforcent leur position précisément parce qu'ils convergent vers une architecture commune.

16.4 La station de travail

Le PC n'est toutefois pas le seul changement qui menace les positions traditionnelles de **DEC**.

Dans les laboratoires, les universités et les bureaux d'études apparaît une autre catégorie de machines : la station de travail.

Des entreprises comme **Sun Microsystems** ou **Apollo Computer** proposent des ordinateurs personnels puissants, équipés d'interfaces graphiques et reliés en réseau. Ces machines sont particulièrement adaptées au développement logiciel, à la simulation scientifique et à la conception assistée par ordinateur.

Elles visent donc précisément une partie du public qui avait historiquement fait le succès de **DEC** : ingénieurs, chercheurs et programmeurs.

La réponse de l'entreprise prend notamment la forme des VAXstations.

Celles-ci bénéficient de l'architecture VAX, de l'expérience de **DEC** dans les réseaux et de l'important environnement logiciel développé autour de VMS.

Mais le marché des stations de travail introduit lui aussi de nouvelles règles.

UNIX y occupe une place croissante. Il permet aux utilisateurs de retrouver des outils et des méthodes similaires sur des machines provenant de constructeurs différents. Les réseaux Ethernet facilitent les échanges entre systèmes, tandis que les interfaces graphiques deviennent progressivement un élément essentiel du poste de travail scientifique.

DEC développe alors ULTRIX, sa propre version d'UNIX.

L'entreprise se retrouve ainsi à soutenir deux logiques différentes.

Avec VMS, la VAXstation prolonge directement l'univers VAX et la maîtrise historique de **DEC**.

Avec ULTRIX, elle entre dans un environnement beaucoup plus ouvert, où les utilisateurs peuvent comparer plus facilement les machines de plusieurs constructeurs.

Cette ouverture permet d'attirer certains clients, mais elle réduit en même temps leur dépendance à l'égard de la plate-forme **DEC**.

16.5 Le logiciel devient déterminant

Dans le modèle traditionnel de **DEC**, l'architecture matérielle occupe une place centrale. Le constructeur définit le processeur, le système d'exploitation et une grande partie de l'environnement utilisé par le client.

Avec le PC et les stations UNIX, cette relation commence à s'inverser.

De plus en plus souvent, l'utilisateur choisit d'abord les logiciels qu'il souhaite employer. Le matériel doit ensuite être capable de les exécuter.

Un service peut par exemple vouloir utiliser un tableur, un traitement de texte ou un logiciel de conception disponible sur une plate-forme devenue standard. La décision d'achat ne porte alors plus uniquement sur les performances du processeur ou sur

l'élégance de l'architecture.

Elle porte sur la compatibilité.

Un ordinateur légèrement moins performant peut devenir préférable s'il donne accès à davantage d'applications, de périphériques et de fournisseurs.

Cette transformation affaiblit l'un des principaux avantages historiques de **DEC** : la qualité de son intégration technique.

Un VAX peut être une machine remarquable. Mais si un ensemble de PC ou de stations de travail moins coûteuses permet d'utiliser les logiciels recherchés tout en laissant davantage de choix au client, la supériorité technique du système ne suffit plus nécessairement.

16.6 Quand l'utilisateur choisit l'écosystème

Le changement touche également la manière dont l'informatique est achetée à l'intérieur des organisations.

L'installation d'un mini-ordinateur ou d'un VAX implique souvent une décision importante. La machine est destinée à plusieurs utilisateurs, nécessite des périphériques spécifiques et s'inscrit dans une infrastructure gérée par un service informatique.

Le PC modifie cette organisation.

Un département peut acheter quelques machines sans engager une transformation complète de son système informatique. Les utilisateurs installent des applications adaptées à leurs besoins et échangent progressivement leurs données à l'aide de formats devenus courants.

L'informatique se décentralise davantage encore.

Cette évolution aurait pu sembler familière à **DEC**. Après tout, l'entreprise avait précisément construit son succès en éloignant le calcul des grands centres informatiques pour le rapprocher des utilisateurs.

Mais la décentralisation des années 1980 se produit dans des conditions très différentes.

Cette fois, l'utilisateur ne choisit plus nécessairement une architecture complète conçue par un seul constructeur. Il choisit un ensemble de logiciels, de périphériques et de standards auxquels

plusieurs constructeurs peuvent donner accès.

Le contrôle se déplace donc du fabricant de l'ordinateur vers l'écosystème.

Pour **DEC**, accepter pleinement cette évolution revient à renoncer à une partie du modèle qui avait fait sa force. Les architectures PDP puis VAX avaient permis à l'entreprise de maîtriser une grande partie de la chaîne technique. Dans le nouveau monde, les processeurs peuvent venir d'un fournisseur, les systèmes d'exploitation d'un autre et les applications d'une multitude d'éditeurs indépendants.

L'entreprise comprend progressivement cette évolution. Elle adopte des protocoles ouverts, développe ULTRIX, intègre TCP/IP et cherche à rendre ses machines compatibles avec des environnements de plus en plus variés.

Mais chacune de ces ouvertures crée une tension avec les activités existantes.

VMS représente un patrimoine considérable et une base de clients très rentable. UNIX attire les ingénieurs et les universités. MS-DOS puis Windows deviennent incontournables sur les ordinateurs personnels.

Soutenir simultanément ces différents mondes exige des investissements importants. En abandonner un signifie risquer de perdre des clients.

16.7 Une force devenue contrainte

À la fin des années 1980, la difficulté de **DEC** ne peut donc pas être réduite à l'échec de quelques ordinateurs personnels.

Le problème est plus profond.

Pendant trois décennies, l'entreprise a construit sa réussite sur plusieurs principes : une forte maîtrise technique, des architectures conçues pour durer, une relation étroite avec les clients et la capacité de proposer des systèmes complets.

Ces principes avaient permis l'apparition des PDP puis du VAX.

Mais le marché évolue désormais vers une autre organisation. Les composants deviennent plus standardisés, les logiciels prennent

une importance croissante et les clients acceptent plus facilement de combiner les produits de plusieurs fournisseurs.

La taille même de **DEC** complique alors son adaptation. L'entreprise compte désormais de très nombreux groupes, plusieurs familles de produits et des bases installées qu'elle ne peut abandonner sans conséquences.

Ce qui avait autrefois favorisé l'innovation peut désormais produire de l'inertie.

Le paradoxe est particulièrement frappant.

DEC avait contribué à libérer l'informatique du modèle centralisé des grands ordinateurs. Elle avait donné aux ingénieurs et aux programmeurs un accès plus direct à la machine.

Mais lorsque cette logique atteint son terme avec l'ordinateur personnel et la station de travail, elle profite à un marché organisé autour de standards que **DEC** ne contrôle plus.

L'informatique personnelle n'échappe donc pas à l'entreprise parce qu'elle aurait ignoré l'idée de rapprocher l'ordinateur de l'utilisateur.

Elle lui échappe parce que cette proximité repose désormais sur une autre règle : l'utilisateur ne choisit plus seulement une machine.

Il choisit un écosystème.

Cinquième partie

Le dernier pari

Chapitre 17

Une culture devenue empire

La culture de **DEC** ne se résume pas à quelques habitudes sympathiques d'une jeune entreprise. Elle forme un système de décision. Une démonstration technique, un prototype qui fonctionne ou l'avis d'un client ingénieur peuvent peser davantage qu'une longue étude de marché. Les équipes reçoivent une autonomie importante et les responsables de produits sont encouragés à se comporter comme des entrepreneurs au sein de la société.

Cette méthode produit des résultats remarquables lorsqu'un marché nouveau reste à inventer. Elle devient plus difficile lorsque l'entreprise doit coordonner des dizaines de gammes, protéger une base installée mondiale et choisir rapidement les produits qu'elle acceptera d'abandonner.

Dans les premières années, l'autonomie ne constitue pas une doctrine écrite. Elle résulte de la petite taille et du manque de ressources. Un ingénieur qui propose une amélioration doit souvent la construire lui-même, trouver des composants et convaincre un client. La croissance transforme cette nécessité en modèle de management. Les lignes de produits disposent de responsables qui pensent leur activité comme une entreprise presque complète.

Cette structure explique la variété de **DEC**. Une équipe peut poursuivre les 36 bits pendant qu'une autre réduit le PDP-8 ou prépare le PDP-11. Les marchés ne sont pas obligés d'attendre qu'une direction centrale comprenne toutes leurs particularités. La concurrence interne produit des idées et attire des personnalités capables de défendre une vision.

Mais elle crée une ambiguïté sur la décision finale. Lorsqu'un projet fonctionne, l'autonomie paraît géniale. Lorsqu'il faut fermer une gamme ou choisir un standard commun, chaque groupe présente des clients, des revenus et une feuille de route. L'organisation excelle à commencer ; elle souffre davantage lorsqu'il faut arrêter.

17.1 Des entrepreneurs à l'intérieur de l'entreprise

Au milieu des années 1960, la croissance révèle les limites de l'organisation informelle des débuts. Les équipes d'ingénierie, la fabrication et les ventes ne se coordonnent pas toujours. *Ken Olsen* met en place une organisation par lignes de produits. Chaque responsable défend son projet, négocie l'accès aux ressources communes et répond de ses résultats devant des comités internes.

Le système évite les frontières rigides de divisions entièrement séparées. Un groupe peut utiliser les mêmes usines, les mêmes vendeurs et les mêmes services qu'un autre. Il favorise l'initiative et permet à plusieurs marchés d'être explorés en parallèle. Le PDP-8, les machines 36 bits, les modules et les systèmes spécialisés peuvent progresser sans attendre une stratégie unique parfaitement ordonnée.

La contrepartie apparaît peu à peu. Les projets entrent en concurrence pour les mêmes ingénieurs et les mêmes budgets. Une équipe peut continuer à défendre une architecture parce qu'elle possède des clients fidèles, même lorsque l'entreprise aurait besoin d'une concentration plus nette. Le débat technique, source de qualité, devient parfois une difficulté à décider.

La culture quotidienne repose aussi sur une forme de franchise. Les réunions peuvent être rudes, les démonstrations servent à contester une décision et le rang hiérarchique ne suffit pas toujours à clore un débat technique. Pour de jeunes ingénieurs, cette atmosphère est stimulante. Elle donne l'impression que la qualité de l'idée compte réellement.

Cette liberté n'est pourtant pas également répartie. Les personnes capables de s'exprimer dans ce langage de confrontation prennent davantage de place, et les décisions restent fortement dépendantes de l'attention des dirigeants. À mesure que la société s'internationalise, un modèle né dans les couloirs de Maynard devient plus difficile à transmettre.

17.2 Le départ d'Harlan Anderson

Harlan Anderson quitte **DEC** en 1966. Les sources s'accordent sur la date, mais attribuent des poids différents aux désaccords personnels, à la réorganisation et aux difficultés de la ligne PDP-6. Il serait réducteur de transformer cette séparation en scène unique. Elle marque néanmoins la fin de la direction à deux qui avait lancé l'entreprise.

Anderson devient ensuite conseiller technologique de **Time Inc.**, puis investisseur dans de jeunes sociétés. Son parcours prolonge l'un des aspects de son rôle initial : relier la technique au financement et repérer des domaines émergents. Olsen, lui, reste à la tête de **DEC** et imprime de plus en plus directement sa personnalité à l'organisation.¹

Cette divergence rappelle que les entreprises ne sont pas seulement conduites par des idées. Elles dépendent de relations, de responsabilités et de la manière dont le pouvoir se distribue lorsque la petite équipe devient une institution.

Le départ d'Anderson prive l'entreprise d'une voix qui avait accompagné le financement et les relations extérieures des débuts. Olsen demeure entouré de dirigeants expérimentés, mais le récit de **DEC** se confond de plus en plus avec le sien. Cette personnalisation renforce l'identité de la société ; elle rend aussi la succession difficile à penser.

Anderson ne disparaît pas du monde technologique. Son activité de conseil et d'investissement lui permet d'observer d'autres entreprises, notamment au moment où les semi-conducteurs et les logiciels créent de nouveaux marchés. Sa trajectoire forme un contrepoint à celle d'Olsen : l'un reste au centre d'un constructeur intégré, l'autre circule entre plusieurs projets.

1. GRAINGER COLLEGE OF ENGINEERING. *Harlan E. Anderson*. University of Illinois Urbana-Champaign ; Edgar H. SCHEIN et al. *DEC Is Dead, Long Live DEC : The Lasting Legacy of Digital Equipment Corporation*. San Francisco : Berrett-Koehler, 2004. ISBN : 978-1-57675-305-7.

17.3 Ken et la preuve technique

Les employés appellent souvent le président simplement « Ken ». La familiarité ne diminue pas son autorité. Olsen circule, pose des questions et ramène volontiers une discussion au besoin réel du client. Il se méfie des effets de mode et valorise les solutions utiles, réparables et économiquement justifiables.

Ce style peut libérer une équipe. Un ingénieur convaincu obtient parfois la possibilité de construire un prototype plutôt que de remplir une longue chaîne d'autorisations. Mais il peut également rendre les décisions imprévisibles. Un projet reçoit un soutien parce qu'il séduit le dirigeant, tandis qu'un autre est interrompu malgré des mois de travail. À mesure que **DEC** grandit, la culture de la discussion directe devient difficile à reproduire dans toutes les filiales.

La politique longtemps associée à l'absence de licenciements renforce la loyauté. Elle encourage les salariés à envisager une carrière entière dans l'entreprise et permet de conserver un savoir technique profond. Lorsque les pertes obligent finalement à supprimer des emplois, le choc est donc culturel autant que financier.

Olsen peut consacrer une attention impressionnante à un détail de produit. Cette qualité protège **DEC** contre certaines décisions purement financières et rappelle aux équipes que le client rencontrera une machine réelle. Mais le dirigeant doit désormais arbitrer des questions qui ne se démontrent pas par un prototype : évolution des canaux de vente, pouvoir des éditeurs de logiciels, effet des standards et vitesse de diffusion d'un écosystème.

La preuve technique reste nécessaire, mais elle n'est plus suffisante. Un produit peut fonctionner admirablement et perdre parce que ses applications arrivent trop tard ou que son prix ne bénéficie pas des mêmes volumes. La culture de l'ingénieur doit apprendre à reconnaître ces mécanismes sans les réduire à des modes.

17.4 L'utilisateur au centre...

Les ingénieurs de **DEC** parlent avec des utilisateurs capables de lire un schéma ou de décrire précisément un problème logiciel.

Cette proximité produit les interfaces ouvertes, les handbooks et les bibliothèques DECUS. Elle donne aussi l'habitude de servir un client qui choisit sa machine pour ses qualités techniques.

Le marché du PC et des standards ouverts fonctionne autrement. L'acheteur peut privilégier le prix, la disponibilité d'une application ou la conformité à une norme, même si la machine est moins élégante. La vente se décide parfois loin de l'ingénieur. La culture de **DEC** interprète trop souvent cette évolution comme une mode passagère ou comme un problème que de meilleurs produits finiront par résoudre.

Au sommet, vers la fin des années 1980, l'entreprise atteint près de 14 milliards de dollars de chiffre d'affaires et plus de 120 000 employés. Elle est devenue, derrière IBM, l'un des plus grands constructeurs du monde.² Sa taille rend désormais toute réorientation extrêmement coûteuse.

La prochaine architecture devra donc accomplir plus qu'une prouesse technique. Elle devra remplacer VAX, préserver plusieurs systèmes d'exploitation, convaincre les développeurs et redonner un avenir à une organisation en crise. Le projet Alpha répond avec éclat à la première partie du problème. Le temps lui manquera pour résoudre les autres.

2. Sara LOTT. *What the DEC ??? Records of Minicomputer Giant Digital Equipment Corporation Open for Research at CHM*. 2017.

Chapitre 18

Alpha contre le temps

Au milieu des années 1980, les ingénieurs de **DEC** voient arriver les architectures RISC.¹ Les stations de travail fondées sur ces processeurs progressent plus vite que les grands jeux d'instructions complexes. VAX, admirablement compatible, porte aussi le poids de décisions prises dans les années 1970.

Plusieurs projets internes cherchent une nouvelle voie. PRISM explore une architecture RISC, mais il est annulé avant d'aboutir à un produit. Les idées et les personnes ne disparaissent pas. Elles nourrissent Alpha, projet lancé avec une ambition exceptionnelle : définir une architecture 64 bits capable de durer plusieurs générations et d'accueillir les principaux systèmes d'exploitation de l'entreprise.

L'annulation de PRISM laisse une blessure et un réservoir d'idées. Des ingénieurs ont travaillé sur les principes RISC, les compilateurs et les systèmes nécessaires. Lorsque le projet Alpha démarre, ils ne repartent pas de rien. Ils disposent aussi d'une leçon : une architecture nouvelle ne survivra pas si elle reste isolée dans une équipe et si l'entreprise ne prépare pas simultanément les logiciels et les produits.

Alpha est donc conçu avec un horizon inhabituellement long. Les architectes cherchent à éviter les décisions qui bloqueraient la fréquence ou l'extension future. Certaines fonctions complexes sont laissées au logiciel pour que les processeurs puissent progresser avec les technologies de fabrication. Cette sobriété contraste avec VAX, dont le jeu d'instructions avait accumulé des opérations riches.

1. Une architecture *RISC* privilégie un ensemble d'instructions relativement simples, conçues pour être exécutées rapidement et pour faciliter le travail du pipeline du processeur. Le terme ne signifie pas qu'un système complet soit simple.

18.1 Une architecture tournée vers l'avenir

Alpha élimine une grande partie de la complexité visible de VAX. Les instructions sont de longueur régulière, les opérations mémoire sont séparées des calculs et le dessin favorise des fréquences élevées. Le passage à 64 bits donne un espace d'adressage immense selon les besoins de l'époque et prépare des applications scientifiques, des serveurs et de grandes bases de données.

Le premier microprocesseur, le 21064, est présenté en 1992. Ses performances placent **DEC** parmi les concepteurs de processeurs les plus avancés. L'architecture ne se limite pas à une puce : elle doit évoluer par familles, avec des implémentations plus rapides et des systèmes allant de la station au serveur multiprocesseur.²

Le choix technique est clair. La difficulté se trouve dans la migration. Les clients VAX possèdent des programmes parfois anciens et indispensables. **DEC** développe des outils de traduction binaire et porte OpenVMS vers Alpha. Digital UNIX exploite également l'architecture, tandis que **Microsoft** propose Windows NT sur certains systèmes Alpha. Une même puce doit donc soutenir plusieurs écosystèmes qui ne poursuivent pas toujours les mêmes objectifs.

La migration de VAX vers Alpha constitue un exploit moins visible que la fréquence du processeur. Les applications OpenVMS doivent continuer à fonctionner, parfois sans disposer immédiatement d'une version recompilée. Les outils de traduction binaire analysent l'ancien code et produisent une forme exécutable sur la nouvelle architecture. Le résultat ne remplace pas toujours une recompilation, mais il protège les clients pendant la transition.

Cette continuité donne à Alpha un accès immédiat à une base logicielle, mais elle lui impose aussi le poids de plusieurs mondes. OpenVMS défend les installations historiques, Digital UNIX cherche les marchés ouverts et Windows NT doit attirer les applications du PC et du serveur. La même architecture sert des stratégies qui ne se renforcent pas toujours mutuellement.

2. ALPHA ARCHITECTURE COMMITTEE. *Alpha Architecture Reference Manual*. Digital Press, 1992. ISBN : 978-1-55558-098-8. DOI : 10.1016/C2009-0-27261-8; COMPUTER HISTORY MUSEUM. *DEC Announces Alpha Chip Architecture*. 1992.

18.2 La puissance en démonstration

Les stations et serveurs Alpha montrent que **DEC** sait encore transformer une architecture en produits de haut niveau. Les systèmes sont employés dans le calcul scientifique, les bases de données et les services Internet. Lancé en 1995, le moteur de recherche AltaVista fonctionne sur des machines Alpha et sert de vitrine à leur capacité à indexer rapidement un volume considérable de pages.

Cette démonstration rappelle les débuts du PDP-1 : un programme spectaculaire rend une architecture visible. Mais le contexte a changé. *Spacewar!* aidait à définir un marché qui n'existait pas encore. AltaVista apparaît dans un secteur où l'avantage se déplace rapidement vers les services en ligne, les logiciels et les volumes d'utilisateurs. La machine puissante ne garantit plus que son constructeur conservera la valeur créée au-dessus d'elle.

AltaVista donne à Alpha une scène publique. Le moteur répond vite, indexe un Web en croissance et démontre les capacités des serveurs. Comme *Spacewar!* autrefois, le programme rend la puissance visible. Mais **DEC** ne transforme pas cette démonstration en domination durable des services Internet. La valeur d'AltaVista se déplace vers la marque, les utilisateurs et la publicité, domaines éloignés du métier historique du constructeur.

Le contraste résume la difficulté des années 1990. **DEC** sait encore fabriquer l'infrastructure d'un futur, mais ce futur est de moins en moins contrôlé par celui qui construit le processeur. Les plates-formes se gagnent par les développeurs, les partenariats et les volumes. Alpha arrive avec des performances remarquables dans une entreprise qui dispose de moins en moins de temps pour bâtir tout l'écosystème nécessaire.

18.3 Une architecture sans répit

Alpha doit combattre sur plusieurs fronts. Les processeurs x86 progressent en performances et bénéficient d'un écosystème immense. Les architectures RISC de **Sun**, **IBM**, **Hewlett-Packard** et d'autres constructeurs occupent déjà les stations et serveurs.

VAX reste rentable chez les clients installés, ce qui ralentit parfois la transition vers son successeur.

Le projet exige en outre des investissements de fabrication considérables. Concevoir un processeur ne suffit pas : il faut suivre la course aux procédés de gravure, produire en volume et attirer les éditeurs. Pour une entreprise dont les revenus se contractent, le calendrier devient aussi important que la qualité de l'architecture.

Alpha prouve donc deux choses apparemment contradictoires. **DEC** conserve une capacité d'innovation exceptionnelle. Mais l'industrie n'est plus structurée de manière à récompenser automatiquement un constructeur qui possède la meilleure architecture propriétaire. La valeur dépend du logiciel, des partenariats, des volumes et de la vitesse avec laquelle un standard se diffuse.

À mesure que les pertes s'accumulent, la société vend des activités et conclut des accords destinés à financer la transition. L'architecture qui devait assurer plusieurs décennies d'avenir devient l'un des actifs que d'autres entreprises chercheront à acquérir.

Chapitre 19

La fin de l'indépendance

La crise de **DEC** ne commence pas le jour où un produit échoue. Elle résulte d'un déplacement général du marché. Les microprocesseurs standard gagnent en puissance, les stations UNIX réduisent le coût du calcul technique, les PC occupent les bureaux et les réseaux ouverts permettent d'assembler des systèmes de marques différentes. La société continue de vendre des VAX, des services et du stockage, mais la croissance qui avait rendu supportable la diversité interne disparaît.

Au début des années 1990, les retards de produits, les investissements dans de grands systèmes et la pression sur les prix provoquent les premières pertes annuelles de l'histoire de l'entreprise. La tradition d'emploi stable cède devant les suppressions de postes. Pour des salariés qui avaient connu une croissance presque continue, la rupture est profonde.

La crise est ressentie jusque dans les lieux qui symbolisaient la croissance. Des bâtiments ferment, des équipes sont regroupées et la politique d'emploi stable ne peut plus être maintenue. Les licenciements ne représentent pas seulement une réduction de coûts : ils rompent la promesse implicite selon laquelle la compétence accumulée trouverait toujours une place dans l'entreprise.

Les clients observent ces changements avec inquiétude. Leur confiance dépend de la durée du support. Une entreprise qui réduit ses effectifs doit continuer à maintenir des machines installées depuis des années tout en finançant Alpha et les nouveaux services. Chaque économie risque d'affaiblir ce qui reste rentable.

19.1 Le départ de Ken Olsen

Ken Olsen quitte la présidence en 1992 après trente-cinq années à la tête de la société. Son départ ne peut être réduit au jugement

selon lequel un fondateur n'aurait pas compris le PC. Il avait construit une organisation capable de déplacer l'informatique des grands centres vers les départements et les réseaux. Mais la structure qu'il dirigeait peinait à choisir entre la protection de VAX/VMS, l'ouverture vers UNIX, le marché du PC et le pari Alpha.

Robert Palmer prend la direction et engage une restructuration plus dure. Les activités sont réorganisées, des sites ferment et des actifs sont vendus. L'objectif est de concentrer l'entreprise sur les serveurs, les services, le stockage et Alpha. Les décisions améliorent certains résultats, mais réduisent aussi l'intégration qui avait fait l'identité de **DEC**.

La société conclut notamment en 1997 un accord avec **Intel** après un conflit de brevets. Intel reprend des activités de fabrication de semi-conducteurs, tandis que **DEC** obtient des ressources et un partenaire industriel. Alpha continue, mais l'entreprise ne maîtrise plus seule toute sa chaîne technologique.

Le départ d'Olsen marque la fin d'une époque, mais pas un redressement immédiat. *Robert Palmer*, issu de la technologie des semi-conducteurs, doit simplifier une organisation dont les gammes et les responsabilités se chevauchent. Il ferme, vend et regroupe. Ces décisions auraient été presque impensables pendant la croissance, lorsque chaque activité pouvait espérer trouver son marché.

La restructuration améliore certaines performances et recentre **DEC** sur les serveurs, le stockage et les services. Elle réduit en même temps la capacité de raconter l'entreprise comme un ensemble unique. Les activités cédées emportent des personnes, des technologies et parfois des relations avec les clients. Le constructeur intégré devient progressivement un portefeuille d'actifs.

19.2 Ce que Compaq veut acheter

Compaq Computer Corporation s'est imposée dans le PC compatible, puis cherche à devenir un fournisseur complet de systèmes d'entreprise. Elle a besoin de services mondiaux, de stockage, de serveurs puissants et de relations avec les grands comptes. **DEC** possède précisément ces éléments, même si son image publique reste associée à une société en difficulté.

L'accord annoncé en janvier 1998 valorise l'opération à environ 9,6 milliards de dollars en numéraire et en actions. Il s'agit alors de la plus importante acquisition de l'industrie informatique. Compaq met en avant le réseau de services de **DEC**, Alpha, OpenVMS, Digital UNIX, les systèmes Windows NT et le stockage.¹

Le rachat est achevé en juin 1998. Quarante et un ans après la fondation de Maynard, **DEC** cesse d'exister comme entreprise indépendante. La marque, les produits et les équipes ne disparaissent pas immédiatement. Ils entrent dans une organisation qui devra elle-même fusionner avec **Hewlett-Packard** en 2002.

Compaq ne rachète donc pas seulement une marque affaiblie. Elle acquiert une présence mondiale dans les services, une technologie de stockage, des serveurs, des systèmes d'exploitation et des ingénieurs capables de travailler avec les grands comptes. L'opération doit transformer un champion du PC en fournisseur d'infrastructure.

Les cultures sont très différentes. Compaq a grandi avec des cycles rapides, des composants standard et une discipline de coûts liée au marché du compatible. **DEC** porte des engagements de support, des architectures propriétaires et une tradition d'ingénierie de système. Les réunir ne consiste pas à ajouter leurs chiffres d'affaires. Il faut décider quelles technologies survivront et comment les clients seront accompagnés.

19.3 Une fin plus complexe qu'un échec

Il serait tentant de raconter la disparition de **DEC** comme la punition d'une entreprise trop attachée à ses mini-ordinateurs. Cette explication contient une part de vérité, mais elle oublie l'ampleur de la transformation industrielle. Les composants standard, les logiciels indépendants et les réseaux ouverts déplacent la valeur hors du constructeur intégré. IBM elle-même doit se réorganiser profondément. D'autres fabricants de mini-ordinateurs disparaissent ou sont absorbés.

1. Elizabeth CORCORAN. *Computer Industry Has a New Giant : Compaq to Buy Digital*. The Washington Post. 27 jan. 1998 ; COMPAQ COMPUTER CORPORATION. *Quarterly Report : Acquisition and Divestiture Disclosures*. U.S. Securities et Exchange Commission, 1999.

DEC commet des erreurs : gammes dispersées, choix tardifs, coûts élevés et difficulté à transformer l'excellence technique en standard de masse. Elle porte aussi le poids de son succès. Des milliers de clients attendent la continuité de systèmes essentiels. Une jeune entreprise peut abandonner un produit ; un constructeur mondial doit fournir des migrations, du support et des pièces pendant des années.

Le rachat clôt le récit institutionnel, mais pas celui des produits. Des machines continuent à fonctionner, des contrats restent actifs et les équipes passent sous de nouveaux noms. Alpha, OpenVMS et les services connaîtront encore plusieurs changements de propriétaires. Cette continuité fragmentée explique pourquoi la disparition de la société ne se traduit pas par une date unique dans la vie des utilisateurs.

La fin de l'indépendance ne supprime donc pas l'héritage. Elle le disperse. Des ingénieurs rejoignent d'autres sociétés. OpenVMS poursuit sa vie. Les concepts de Windows NT portent l'expérience de *Dave Cutler*. Les terminaux, les réseaux, UNIX, C et les architectures issues des PDP continuent à influencer les systèmes. Même le mot « Digital », devenu banal, garde quelque chose de l'époque où une petite entreprise choisit ce nom pour éviter d'écrire « computer » sur son premier projet.

Ce que DEC a rendu possible

L'histoire de **DEC** commence par une réduction : réduire la taille, le prix et la distance qui séparent l'utilisateur de l'ordinateur. Elle ne s'achève pas lorsque cette réduction atteint le micro-ordinateur. Entre 1957 et 1998, l'entreprise a contribué à modifier plusieurs fois la place de la machine.

Le PDP-1 montre qu'un ordinateur commercial peut être interactif. Le PDP-8 fait du calculateur un composant de laboratoire ou d'équipement. Le PDP-10 donne une forme sociale au temps partagé. Le PDP-11 fournit une architecture régulière autour de laquelle grandissent systèmes, périphériques et langages. VAX et VMS protègent le logiciel au moment où les machines deviennent des plates-formes. DECnet et VAXcluster transforment plusieurs ordinateurs en environnement commun. Alpha rappelle enfin que l'innovation technique peut survivre plus longtemps que le modèle économique qui la porte.

Mais l'héritage le plus profond se trouve peut-être dans les pratiques. **DEC** a souvent considéré l'utilisateur comme une personne techniquement compétente, capable de lire un manuel, d'écrire une interface et de proposer une amélioration. Cette confiance favorise les communautés, les bibliothèques DECUS et les usages imprévus. Elle explique pourquoi des machines produites à quelques dizaines d'exemplaires ont laissé une trace disproportionnée.

La trajectoire des personnes prolonge celle des machines. *Gordon Bell* apporte son expérience de l'architecture à d'autres entreprises et à la recherche. *Dave Cutler* dirige le développement de Windows NT. Les programmeurs formés sur UNIX et C bâtissent des systèmes sur des processeurs qui n'existaient pas lorsque le PDP-11 fut dessiné. Les idées ne restent pas enfermées dans les produits qui les ont fait naître.

Il en va de même des interfaces. Les séquences du VT100 vivent dans les émulateurs de terminaux, les principes des bus réapparaissent sous d'autres formes et les clusters annoncent des services

distribués où la panne d'une machine ne doit pas interrompre l'ensemble. Ces héritages ne sont pas des copies directes. Ils montrent comment une solution historique peut devenir une habitude de pensée.

Les systèmes de **DEC** restent aujourd'hui accessibles par les collections, les restaurations et les simulateurs. Le projet SIMH, initié par *Bob Supnik*, permet d'exécuter de nombreux environnements historiques sur du matériel contemporain. Les répliques physiques et les restaurations du Computer History Museum redonnent aux panneaux, aux rubans et aux terminaux une présence que les photographies ne peuvent restituer.²

Cette conservation ne relève pas seulement de la nostalgie. Elle permet de comprendre les contraintes qui ont façonné les architectures et les logiciels. Programmer avec quelques milliers de mots, attendre un télétype ou charger un système depuis DECtape oblige à regarder autrement les abstractions modernes. L'histoire technique devient une expérience.

La conservation restitue aussi la lenteur et la matérialité. Un manuel décrit une commande, mais il ne reproduit pas le bruit du télétype, l'attente d'un ruban ou l'attention portée aux lampes du panneau. Les restaurations et les répliques ne remplacent pas les archives ; elles ajoutent l'expérience des contraintes.

Pour l'historien, cette expérience évite une illusion fréquente : juger les anciennes machines à partir des possibilités modernes. Une mémoire de quelques milliers de mots n'est pas seulement une quantité faible. Elle structure le programme, impose des choix et rend certaines solutions élégantes. Comprendre ces limites permet de comprendre les personnes qui ont travaillé avec elles.

DEC n'a pas inventé seule l'informatique interactive, le mini-ordinateur, UNIX, le réseau ou la station de travail. Aucune entreprise ne possède une telle histoire à elle seule. Son rôle fut de relier des idées, de les transformer en produits et de les placer entre les mains de communautés qui leur donnèrent une portée nouvelle.

C'est pourquoi l'ordre de ce récit compte. Les modules conduisent

2. OPEN SIMH PROJECT. *Open SIMH* ; COMPUTER HISTORY MUSEUM. *PDP-1 Restoration Project*.

au PDP-1 ; le PDP-1 au MIT ; le MIT révèle la puissance d'une communauté ; le PDP-8 élargit le marché ; les périphériques transforment les usages ; le PDP-11 rend possible un écosystème ; UNIX et C dépassent cet écosystème ; VAX tente de préserver la continuité ; les réseaux ouvrent le système ; cette ouverture finit par rendre plus difficile le modèle intégré de **DEC**.

L'entreprise disparaît, mais la question qui l'avait fondée demeure : comment rapprocher la puissance de calcul des personnes qui en ont besoin ? Chaque génération d'informatique apporte une réponse nouvelle. Pendant quatre décennies, **DEC** fut l'un des lieux où cette réponse s'inventa.

Chapitre A

Chronologie

Cette chronologie rassemble les principaux jalons du récit. Elle ne cherche pas à recenser chaque modèle ni chaque version logicielle. Les dates correspondent aux annonces ou aux premières livraisons lorsqu'elles sont clairement établies par les archives de **DEC** et les documents techniques.

Date	Événement
1951	Création du MIT Lincoln Laboratory. Les travaux sur la défense aérienne, le temps réel et les calculateurs interactifs forment l'environnement dans lequel travailleront <i>Ken Olsen</i> et <i>Harlan Anderson</i> .
1952	Anderson rejoint le Lincoln Laboratory, où Olsen devient son premier responsable.
1957	Fondation de Digital Equipment Corporation à Maynard, dans un ancien moulin. American Research and Development apporte le financement initial.
1957	Commercialisation des premiers <i>Laboratory Modules</i> , destinés aux ingénieurs, aux laboratoires et aux systèmes de contrôle.
1958	Introduction des <i>System Modules</i> , plus denses et adaptés à des ensembles permanents.
1959	Présentation publique du prototype PDP-1 à l'Eastern Joint Computer Conference.
1960	Livraison du premier PDP-1 de production à Bolt Beranek and Newman .
1960	<i>Gordon Bell</i> rejoint DEC et participe aux extensions du PDP-1, puis aux architectures suivantes.
1961	Don du deuxième PDP-1 de production au Research Laboratory for Electronics du MIT.

Date	Événement
1962	Développement et diffusion de <i>Spacewar!</i> sur le PDP-1. Des expériences de temps partagé sont menées au MIT et chez BBN.
1963	Introduction du PDP-5, première architecture 12 bits de la société destinée à un marché économique de laboratoire et de contrôle.
1964	Annnonce du PDP-6, machine 36 bits conçue pour le calcul scientifique et le temps partagé.
1965	Le PDP-8 élargit le marché du mini-ordinateur grâce à un prix de base annoncé à 18 000 dollars pour une configuration exploitable.
1965	Création de DECUS, association d'utilisateurs favorisant l'échange de programmes et d'expériences.
1966	Départ de <i>Harlan Anderson</i> . <i>Ken Olsen</i> demeure à la tête de l'entreprise.
1967	Lancement du PDP-10, compatible avec le PDP-6 et mieux adapté à la diffusion commerciale du temps partagé.
1968	Création de Data General par plusieurs anciens de DEC , dont <i>Edson de Castro</i> .
1970	Livraison du PDP-11/20, premier modèle d'une famille 16 bits caractérisée par une architecture régulière et l'Unibus.
1971	UNIX devient opérationnel sur PDP-11 aux Bell Laboratories et commence à être utilisé pour le traitement de documents.
1973	Le noyau d'UNIX est réécrit en grande partie en langage C, renforçant sa portabilité.
1975	Annnonce des premières phases de DECnet, architecture destinée à relier plusieurs systèmes DEC .
1975	Début des travaux du comité VAX sur une extension 32 bits culturellement compatible avec le PDP-11.
1977	Présentation du VAX-11/780 et de l'architecture VAX.
1978	Livraison de VMS 1.0. Introduction du terminal VT100, dont les séquences de contrôle deviendront une référence durable.

Date	Événement
1978	Publication de <i>The C Programming Language</i> par <i>Brian Kernighan</i> et <i>Dennis Ritchie</i> .
1980	Coopération entre DEC , Intel et Xerox autour de la spécification Ethernet commerciale.
1981	Lancement de l'IBM PC, qui accélère la formation d'un standard de bureau fondé sur des composants et logiciels indépendants.
1982– 1983	DEC propose plusieurs réponses au poste personnel, notamment le DECmate, le Professional 300 et le Rainbow 100.
1983	Annnonce des VAXclusters, qui associent plusieurs VAX et un stockage partagé sous VMS.
1984	Introduction d'ULTRIX-32, version UNIX de DEC pour la famille VAX.
1988	<i>Dave Cutler</i> quitte DEC et rejoint Microsoft , où il dirigera le développement de Windows NT.
1990	L'entreprise connaît la première perte annuelle de son histoire après plus de trois décennies de croissance.
1992	Départ de <i>Ken Olsen</i> . <i>Robert Palmer</i> prend la direction de la société.
1992	Présentation du microprocesseur Alpha 21064 et des premiers systèmes fondés sur l'architecture 64 bits Alpha.
1995	Lancement d'AltaVista sur des serveurs Alpha, démonstration publique de leur capacité à traiter rapidement un Web en forte croissance.
1997	Accord entre DEC et Intel à la suite d'un conflit de brevets ; Intel reprend notamment des activités de fabrication de semi-conducteurs.
1998	Annnonce puis achèvement du rachat de DEC par Compaq . L'entreprise cesse d'exister comme constructeur indépendant.
2002	Rachat de Compaq par Hewlett-Packard . Plusieurs technologies et équipes issues de DEC changent à nouveau de cadre industriel.

Chapitre B

Repères sur les familles

Le nom PDP ne désigne pas une architecture unique. **DEC** l'emploie pour plusieurs lignées incompatibles, définies selon les marchés et les technologies disponibles. Le tableau suivant fournit des repères, sans remplacer les explications du récit.

Famille	Repères	Orientation principale
PDP-1	18 bits 1960	Informatique interactive, laboratoires, affichage graphique, recherche et premiers usages en temps partagé.
PDP-4, PDP-7, PDP-9, PDP-15	18 bits 1962	Prolongement plus économique et plus industrialisé de la lignée 18 bits pour le calcul scientifique, le contrôle et l'instrumentation.
PDP-5, PDP-8, PDP-12	12 bits 1963	Systèmes économiques de laboratoire, d'enseignement, de contrôle et d'intégration OEM. Le PDP-12 associe notamment les traditions du LINC et du PDP-8.
PDP-6, PDP-10, DECsystem-10, DECSYSTEM-20	36 bits 1964	Calcul scientifique, intelligence artificielle, universités, centres de recherche et temps partagé multiutilisateur.
PDP-11	16 bits 1970	Famille générale allant du contrôle embarqué aux systèmes multiutilisateurs, avec une architecture régulière et un vaste écosystème de périphériques.
VAX	32 bits 1977	Extension de l'héritage PDP-11 vers la mémoire virtuelle, les applications plus importantes et une gamme allant du MicroVAX aux systèmes d'entreprise.
Alpha	64 bits 1992	Architecture RISC à hautes performances pour stations, serveurs, OpenVMS, Digital UNIX et certains systèmes Windows NT.

Quelques modèles structurants

PDP-1 Premier ordinateur complet commercialisé par **DEC**. Sa portée historique vient de l'interaction directe, de l'affichage et de l'environnement créé autour de lui au MIT.

PDP-8 Machine emblématique de la réduction des coûts. Ses nombreuses variantes maintiennent une architecture 12 bits pendant plusieurs générations technologiques.

PDP-10 Expression commerciale la plus durable de la lignée 36 bits. Il accueille des communautés de temps partagé et plusieurs travaux importants en intelligence artificielle et en réseau.

PDP-11/20 Premier modèle 16 bits de la famille PDP-11. Il introduit l'Unibus et une architecture dont la régularité favorisera les compilateurs et les systèmes d'exploitation.

VAX-11/780 Première réalisation de l'architecture VAX. Son nom manifeste la continuité avec le PDP-11, tandis que ses adresses 32 bits et VMS ouvrent une nouvelle génération de systèmes.

MicroVAX Déclinaison plus compacte et plus accessible de l'environnement VAX, destinée aux départements, aux laboratoires et aux réseaux distribués.

Alpha 21064 Première implémentation commerciale d'Alpha. Elle démontre que **DEC** reste capable de concevoir un microprocesseur parmi les plus performants de son époque.

Les dates et catégories doivent être lues comme des repères. Une annonce peut précéder la livraison, et un même modèle peut servir des usages très différents selon ses périphériques et son système d'exploitation.

Chapitre C

Glossaire

Adresse Nombre qui désigne une position de mémoire ou, dans certaines architectures, un registre de périphérique. La largeur des adresses limite l'espace directement accessible par un programme.

Architecture Ensemble des caractéristiques visibles par le programmeur : instructions, registres, types de données, adressage et organisation des interruptions. Deux machines physiquement différentes peuvent partager la même architecture et exécuter les mêmes programmes.

Assembleur Langage qui représente les instructions de la machine par des noms symboliques. Le même terme désigne aussi le programme qui traduit ce texte en code exécutable.

Bus Ensemble de lignes et de règles permettant à plusieurs composants d'échanger des adresses, des données et des signaux de contrôle. L'Unibus du PDP-11 relie processeur, mémoire et périphériques selon un modèle commun.

Cluster Groupe de machines qui coopèrent pour fournir des services ou partager des ressources. Dans un VAXcluster, plusieurs systèmes VMS peuvent notamment accéder à un stockage commun et coordonner leurs fichiers.

Compatibilité Capacité à préserver des programmes, des données, des périphériques ou des habitudes entre plusieurs modèles. La compatibilité peut être binaire, logicielle, matérielle ou seulement « culturelle », comme lors du passage du PDP-11 à VAX.

DECnet Architecture de réseau développée par DEC à partir du milieu des années 1970. Elle relie plusieurs familles de machines et de systèmes d'exploitation avant de devoir coexister avec Ethernet, TCP/IP et les normes multiconstructeurs.

DECTape Support magnétique à petites bobines organisé en blocs adressables. Il offre sur plusieurs systèmes DEC un usage

plus souple que la bande séquentielle traditionnelle.

DECUS Association d'utilisateurs de **DEC**, créée au milieu des années 1960. Elle organise l'échange de logiciels, les réunions techniques et la représentation des besoins des utilisateurs.

Entrée-sortie Ensemble des opérations par lesquelles l'ordinateur reçoit des données ou agit sur l'extérieur : clavier, télétype, disque, écran, capteur, réseau ou relais industriel.

Interruptions Mécanisme permettant à un événement matériel ou logiciel de suspendre momentanément le programme courant afin d'exécuter une routine particulière. Il évite au processeur de vérifier continuellement l'état de chaque périphérique.

Logiciel système Programme qui organise les ressources de la machine et fournit un environnement aux applications : système d'exploitation, assembleur, compilateur, éditeur, chargeur ou diagnostic.

Mémoire à tores Mémoire utilisant de petits anneaux magnétiques traversés par des fils. Elle a dominé de nombreux ordinateurs des années 1950 aux années 1970 et conserve généralement son contenu après la coupure de l'alimentation.

Mémoire virtuelle Technique qui sépare les adresses utilisées par un programme des emplacements physiques de la mémoire. Le système peut protéger les processus et déplacer des pages entre mémoire et stockage.

Mini-ordinateur Catégorie historique de machines plus petites et moins coûteuses que les grands systèmes, souvent installées dans un laboratoire, un département ou un équipement industriel. Le terme décrit un marché et un mode d'usage autant qu'une taille physique.

OEM *Original Equipment Manufacturer*. Entreprise qui intègre le processeur ou le système d'un fournisseur dans son propre équipement, par exemple un instrument médical ou une machine industrielle.

Périphérique Équipement connecté au processeur : terminal, lecteur de bande, disque, imprimante, écran, interface réseau ou dispositif d'acquisition. Le périphérique détermine souvent l'usage réel de la machine.

Plate-forme Ensemble relativement stable formé par une architec-

ture, un système, des outils, des périphériques et des conventions. Le terme souligne que la valeur ne réside pas uniquement dans le processeur.

Portabilité Possibilité d'adapter un programme à une autre architecture avec un effort limité. L'écriture d'UNIX en C a montré qu'une grande partie d'un système pouvait survivre au remplacement du processeur.

Processeur Partie de l'ordinateur qui exécute les instructions. Dans les premiers PDP, il occupe plusieurs cartes ou armoires ; dans les générations ultérieures, il peut être intégré dans quelques circuits puis dans un microprocesseur.

RISC Famille de principes d'architecture privilégiant des instructions régulières et relativement simples, adaptées à l'exécution en pipeline. Alpha appartient à cette tradition.

Système d'exploitation Logiciel qui gère la mémoire, les fichiers, les périphériques, les utilisateurs et l'exécution des programmes. OS/8, TOPS-10, RSX-11, UNIX et VMS correspondent à des objectifs et des machines différents.

Temps partagé Organisation dans laquelle le processeur passe rapidement d'une session à l'autre afin de donner à plusieurs utilisateurs une interaction presque simultanée.

Temps réel Fonctionnement dans lequel le système doit répondre avant une échéance imposée par le phénomène observé ou commandé. Le terme ne signifie pas simplement « très rapide ».

Terminal Dispositif de saisie et d'affichage relié à un ordinateur. Les premiers terminaux sont souvent des télétypes mécaniques ; les modèles vidéo comme le VT100 utilisent un écran et des séquences de contrôle.

VMS *Virtual Memory System*, système d'exploitation développé avec VAX. Il devient plus tard OpenVMS et se distingue par son intégration, sa sécurité, ses outils et ses mécanismes de disponibilité.

Bibliographie

- ALPHA ARCHITECTURE COMMITTEE. *Alpha Architecture Reference Manual*. Digital Press, 1992. ISBN : 978-1-55558-098-8. DOI : 10.1016/C2009-0-27261-8.
- CERUZZI, Paul E. *A History of Modern Computing*. 2^e éd. Cambridge, Massachusetts : MIT Press, 2003. ISBN : 978-0-262-53203-7.
- COMPAQ COMPUTER CORPORATION. *Quarterly Report : Acquisition and Divestiture Disclosures*. U.S. Securities et Exchange Commission, 1999. URL : <https://www.sec.gov/Archives/edgar/data/714154/000101540299001323/0001015402-99-001323-d1.html>.
- COMPUTER HISTORY MUSEUM. *Digital Equipment Corporation Founded*. 1957. URL : <https://www.computerhistory.org/timeline/1957/>.
- *Invitation to a Presentation of a PDP Computer to Massachusetts Institute of Technology*. 1961. URL : <https://www.computerhistory.org/pdp-1/e282ffc5d7330dc5747327a33f06afaa/>.
- *Invitation to the Presentation of a PDP-1 to MIT*. 1961. URL : <https://www.computerhistory.org/pdp-1/e282ffc5d7330dc5747327a33f06afaa/>.
- *“Steal from Your Friends” DECUS Button*. 1965. URL : <https://www.computerhistory.org/revolution/mainframe-computers/7/172/697>.
- *VAX-11/780*. 1978. URL : <https://www.computerhistory.org/timeline/1978/>.
- *VT100 Terminal*. 1978. URL : <https://www.computerhistory.org/revolution/input-output/14/349/1697>.
- *Networked Storage Systems Commercialized*. 1983. URL : <https://www.computerhistory.org/storageengine/networked-storage-systems-commercialized/>.
- *DEC Announces Alpha Chip Architecture*. 1992. URL : <https://www.computerhistory.org/timeline/1992/>.

- COMPUTER HISTORY MUSEUM. *Alan Kotok*. URL :
<https://computerhistory.org/profile/alan-kotok/>.
- *Digital Equipment Corporation*. URL : <https://www.computerhistory.org/revolution/minicomputers/11/335>.
- *Ed Fredkin*. URL :
<https://www.computerhistory.org/pdp-1/ed-fredkin/>.
- *Harlan Anderson*. URL :
<https://www.computerhistory.org/pdp-1/harlan-anderson/>.
- *Harlan Anderson*. URL :
<https://computerhistory.org/profile/harlan-anderson/>.
- *Ken Olsen*. URL :
<https://www.computerhistory.org/pdp-1/ken-olsen/>.
- *Origins of the PDP-1*. URL : <https://www.computerhistory.org/pdp-1/origins-of-the-pdp-1/>.
- *PDP-1 Restoration Project*. URL :
<https://www.computerhistory.org/pdp-1/>.
- *Scientific Applications on the PDP-1*. URL :
<https://www.computerhistory.org/pdp-1/>.
- *Spacewar!* URL :
<https://www.computerhistory.org/pdp-1/spacewar/>.
- *The PDP-8*. URL : <https://www.computerhistory.org/revolution/minicomputers/11/331>.
- *Timesharing on the PDP-1*. URL :
<https://www.computerhistory.org/pdp-1/timesharing/>.
- CORCORAN, Elizabeth. *Computer Industry Has a New Giant : Compaq to Buy Digital*. The Washington Post. 27 jan. 1998. URL :
<https://www.washingtonpost.com/archive/politics/1998/01/27/computer-industry-has-a-new-giant/1dc0f99b-c7ba-4cbf-a87d-a609910269fd/>.
- DIGITAL EQUIPMENT CORPORATION. *VT100 User Guide*. Digital Equipment Corporation. 1978. URL :
https://bitsavers.org/pdf/dec/terminal/vt100/EK-VT100-UG-001_VT100_User_Guide_Aug78.pdf.
- *36-bit Systems Timeline*. Digital Computing History Timeline. 1997. URL : https://archive.computerhistory.org/resources/text/DEC/dec.digital_%28DEC%29_timeline_1957-1997.102630354/36-bit.htm.
- *Networks Timeline*. Digital Computing History Timeline. 1997. URL : <https://archive.computerhistory.org/resources/>

- text/DEC/dec.digital_%28DEC%29_timeline_1957-1997.102630354/networks.htm.
- DIGITAL EQUIPMENT CORPORATION, INFORMATION RESEARCH SERVICES. *DIGITAL Computing History Timeline, 1957-1997*. 1998. URL : https://archive.computerhistory.org/resources/text/DEC/dec.digital_%28DEC%29_timeline_1957-1997.102630354/timelinetext.htm.
- GRAINGER COLLEGE OF ENGINEERING. *Harlan E. Anderson*. University of Illinois Urbana-Champaign. URL : <https://grainger.illinois.edu/alumni/hall-of-fame/harlan-anderson>.
- IBM. *The IBM 1401*. IBM Heritage. URL : <https://www.ibm.com/history/1401>.
- *The IBM System/360*. IBM Heritage. URL : <https://www.ibm.com/history/system-360>.
- LEVY, Steven. *Hackers : Heroes of the Computer Revolution*. Garden City, New York : Anchor Press/Doubleday, 1984.
- LOTT, Sara. *What the DEC ? ! ? Records of Minicomputer Giant Digital Equipment Corporation Open for Research at CHM*. 2017. URL : <https://computerhistory.org/blog/what-the-dec-records-of-minicomputer-giant-digital-equipment-corporation-open-for-research-at-chm/>.
- MIT LINCOLN LABORATORY. *Interactive Supercomputing : From Whirlwind to TX-2*. Massachusetts Institute of Technology. URL : <https://www.ll.mit.edu/about/history/interactive-supercomputing>.
- *Lincoln Laboratory History*. Massachusetts Institute of Technology. URL : <https://www.ll.mit.edu/about/history>.
- MIT NEWS. *Kenneth Olsen, Computing Pioneer and Founder of DEC, Dies at 84*. 2011. URL : <https://news.mit.edu/2011/obit-olsen>.
- OPEN SIMH PROJECT. *Open SIMH*. URL : <https://opensimh.org/>.
- RITCHIE, Dennis M. *The Evolution of the Unix Time-Sharing System*. 1984. URL : <https://www.bell-labs.com/usr/dmr/www/hist.html>.
- *The Development of the C Language*. 1993. URL : <https://www.bell-labs.com/usr/dmr/www/chist.html>.
- RITCHIE, Dennis M. et Ken THOMPSON. « The UNIX Time-Sharing System ». In : *Communications of the ACM* 17.7 (1974), p. 365-375. DOI : 10.1145/361011.361061.

- SCHEIN, Edgar H. et al. *DEC Is Dead, Long Live DEC : The Lasting Legacy of Digital Equipment Corporation*. San Francisco : Berrett-Koehler, 2004. ISBN : 978-1-57675-305-7.
- STRECKER, William D. « VAX-11/780 : A Virtual Address Extension to the DEC PDP-11 Family ». In : *AFIPS Conference Proceedings* 47 (1978), p. 967-980. URL : https://gordonbell.azurewebsites.net/Computer_Engineering/00000431.htm.

Index général

- 12 bits, 57
- 16 bits, 91
- 32 bits, 113
- 64 bits, 143
- 21064, 143
- adressage, 91
- affichage graphique, 71
- Alpha, 143
 - famille, 159
- AltaVista, 143
- American Research and Development Corporation, 21
- Harlan Anderson, ix, 9
 - départ, 137
- ARPANET, 83, 88
- assembleur, 47
- bande magnétique, 71
- BASIC, 65
- BBN, *voir* Bolt Beranek and Newman
- Bell Laboratories, 99
- Gordon Bell, 57, 91
- Bolt Beranek and Newman, 41, 44
- byte, 91
- capital-risque, 21
- cartes perforées, 1
- Compaq Computer Corporation, 147, 148
- compatibilité, 113
- compatibilité logicielle, 117
- conservation, 151
- convertisseur analogique-numérique, 71
- crise de Digital Equipment Corporation, 147
- Dave Cutler, 117
- Data General, 91
- DCL, 117
- DDT, 47
- Edson de Castro, 57
- DECmate, 127
- DECnet, 121
- DECsystem-10, 83
- DECSYSTEM-20, 83
- DECtape, 71, 74
- Digital Equipment Computer Users Society, 105, 107
- Jack Dennis, 47, 48
- Digital Building Blocks, 29
- Digital Equipment Corporation, ix
 - chronologie, 155
 - culture, 137
 - fondation, 21
 - héritage, 151
 - universités, 47
- Digital Press, 105
- Digital UNIX, 143

- disque magnétique, 71
documentation technique, 29, 105
Georges Doriot, 21

enseignement, 47
Ethernet, 121

Flip Chip, 57
FOCAL, 65
FORTRAN, 65
Ed Fredkin, 41, 44

glossaire, 161
Ben Gurley, 41, 42

hacker, 47
HSC50, 121

IBM, 1
IBM PC, 127
ILLIAC I, 9
informatique interactive, 15, 41
ingénieurs, 137
instrumentation, 71
Intel, 147
intelligence artificielle, 83

Brian Kernighan, 99, 103
Alan Kotok, 47, 51, 83

Laboratory Modules, 29
langage C, 99
licenciements, 147
logiciel, 105

Maynard, 21
mini-ordinateur, 57
MIT, 9
Research Laboratory for Electronics, 47
MIT Lincoln Laboratory, 15
modularité, 15
modules logiques, 29
Multics, 99
mémoire virtuelle, 113
mémoire à tores, 15

Nova, 91

OEM, 57, 105
Ken Olsen, ix, 9
 départ, 147
 management, 137
OMNIBUS, 57
OpenVMS, 117, 143
ordinateur central, 1
ordinateur personnel, 127
organisation matricielle, 137
OS/8, 105

Robert Palmer, 147, 148
PDP
 familles, 159
PDP-1, 41
 don au MIT, 47
PDP-10, 83, 86
PDP-11, 91
 UNIX, 99
PDP-5, 57
PDP-6, 83
PDP-7, 99
PDP-8, 57
portabilité, 99
PRISM, 143
product line manager, 137
production, 29
Professional 300, 127

- périphériques, 71
- rachat de Digital Equipment Corporation, 147
- Rainbow 100, 127
- Research Laboratory for Electronics, 47
- restructuration, 147
- RISC, 143
- Dennis Ritchie, 99, 100
- RSTS/E, 105
- RSX-11, 105
- RSX-11M, 117
- RT-11, 105
- Steve Russell, 47, 53
- réseau local, 121
- SAGE, 15
- Peter Samson, 47, 54
- service de terrain, 105
- SIMH, 151
- simulation, 151
- Spacewar, 47
- Star, 113
- station de travail, 127
- stockage partagé, 121
- William Strecker, 113, 114
- System Modules, 29
- System/360, 5
- système d'exploitation, 117
- Tech Model Railroad Club, 47
- Teletype Model 33 ASR, 72
- temps partagé, 47, 83, 84
- TENEX, 89
- terminal, 121
- Ken Thompson, 99, 100
- TMRC, *voir* Tech Model Railroad Club
- TOPS-10, 83, 88
- TOPS-20, 83, 89
- traitement par lots, 1
- TU55, 74
- TU56, 74
- TX-0, 15
- TX-2, 15
- Type 30, 41
- télétype, 71
- ULTRIX, 127
- Unibus, 91
- University of Illinois, 9
- UNIX, 99, 127
- utilisateurs, 137
- VAX, 113
- famille, 159
- VAX-11/780, 113
- VAXcluster, 121
- VAXstation, 127
- VMS, 117
- VT100, 121
- Whirlwind, 15
- Windows NT, 143
- Works Committee, 137