Département Informatique - CNAM

- VARI -

Architecture des ordinateurs

Pierre Cubaud
cubaud@cnam.fr

Plan de l'exposé

1. **1838 : Le calculateur universel**
   le processeur

2. **1945 : L'ordinateur**
   la mémoire

3. **1973 : La station de travail**
   les "entrées/sorties"

4. **Un exemple aujourd'hui : le G5**

Bibliographie

• Techniques de l'ingénieur (en ligne depuis le CNAM)

• ZANELLA, LIGIER Architecture et technologie des ordinateurs, Dunod 1993

• PATTERSON, HENNESSY Organisation et conception des ordinateurs, Dunod 1994

• MERCOUROFF Les ordinateurs et les microprocesseurs, Cedic 1986
• ETIEMBLE Les processeurs RISC, A. Colin 1991
• ROSENCHER La puce et l'ordinateur, Dominos Flammarion 1995
• DAUVIN et al. Les composants éléctroniques et leurs industrie QSJ1080, 1995
• LIGONNIERE Prehistoire et histoire des ordinateurs, Laffont 1987
• BIRRIEN Histoire de l'informatique QSJ2510, 1992

Acte I : 1838

Le calcul par tables (logarithmes, trigonométrie, …) :

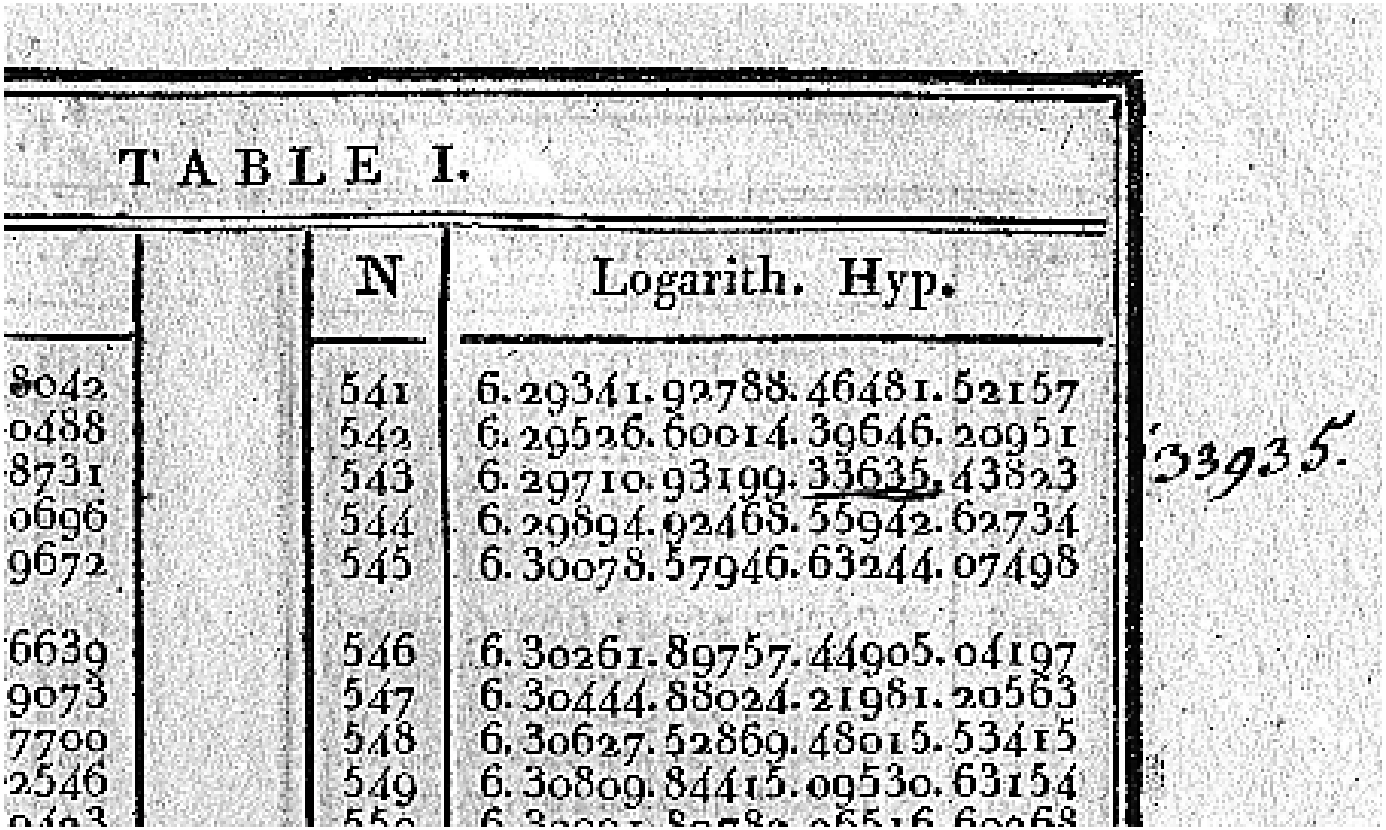


TABLE I.

	N	Logarith. Hyp.
8042	541	6. 29341. 92788. 46481. 52157
0488	542	6. 29526. 60014. 39646. 20951
8731	543	6. 29710. 93199. 33635. 43823
0696	544	6. 29894. 92468. 55942. 62734
9672	545	6. 30078. 57946. 63244. 07498
6639	546	6. 30261. 89757. 44905. 04197
9073	547	6. 30444. 88024. 21981. 20563
7700	548	6. 30627. 52869. 48015. 53415
2546	549	6. 30809. 84415. 09530. 63154

33935.

Tables calculées par la méthode des différences finies :

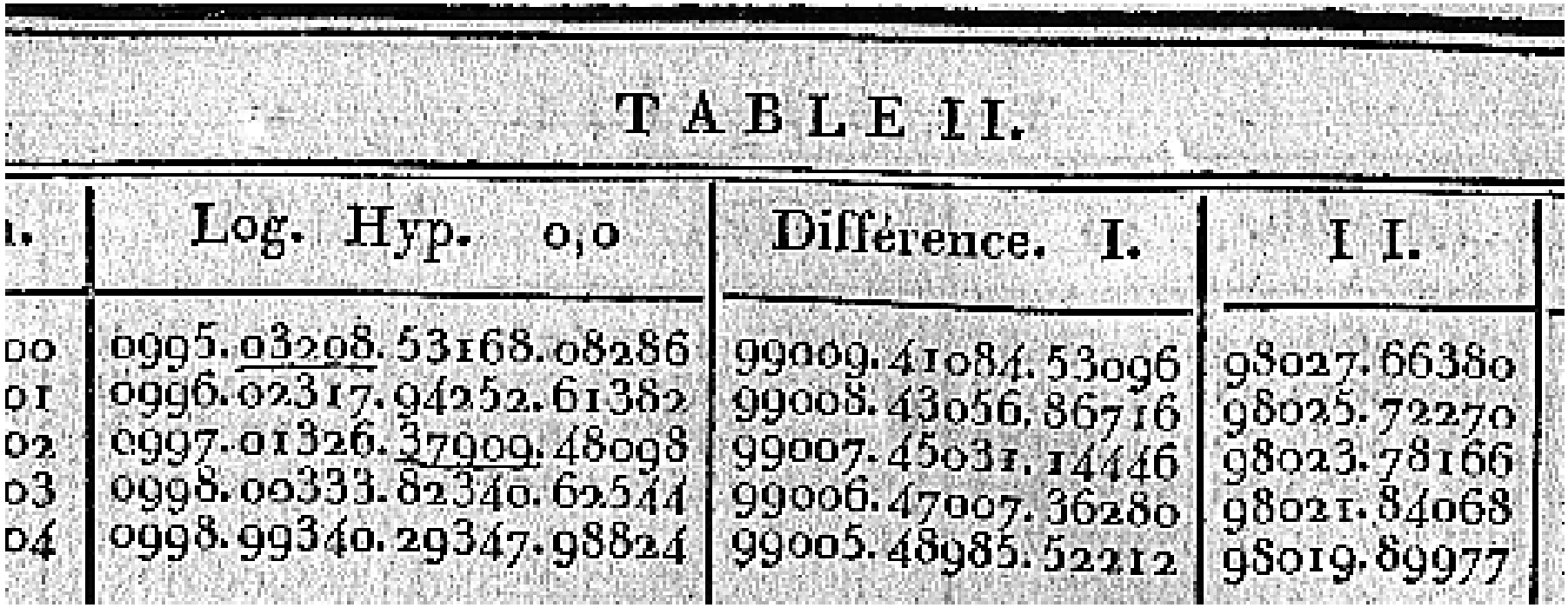
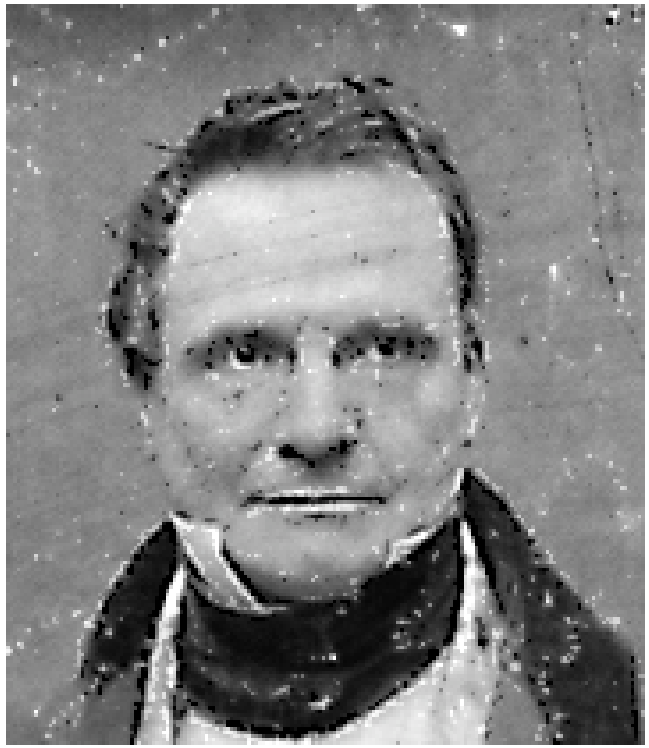


TABLE II.

ı.	Log. Hyp. o,o	Différence. I.	I I.
00	0995.03208.53168.08286	99009.41084.53096	98027.66380
01	0996.02317.94252.61382	99008.43056.86716	98025.72270
02	0997.01326.37909.48098	99007.45031.14446	98023.78166
03	0998.00333.82340.62544	99006.47007.36280	98021.84068
04	0998.99340.29347.98824	99005.48985.52212	98019.89977

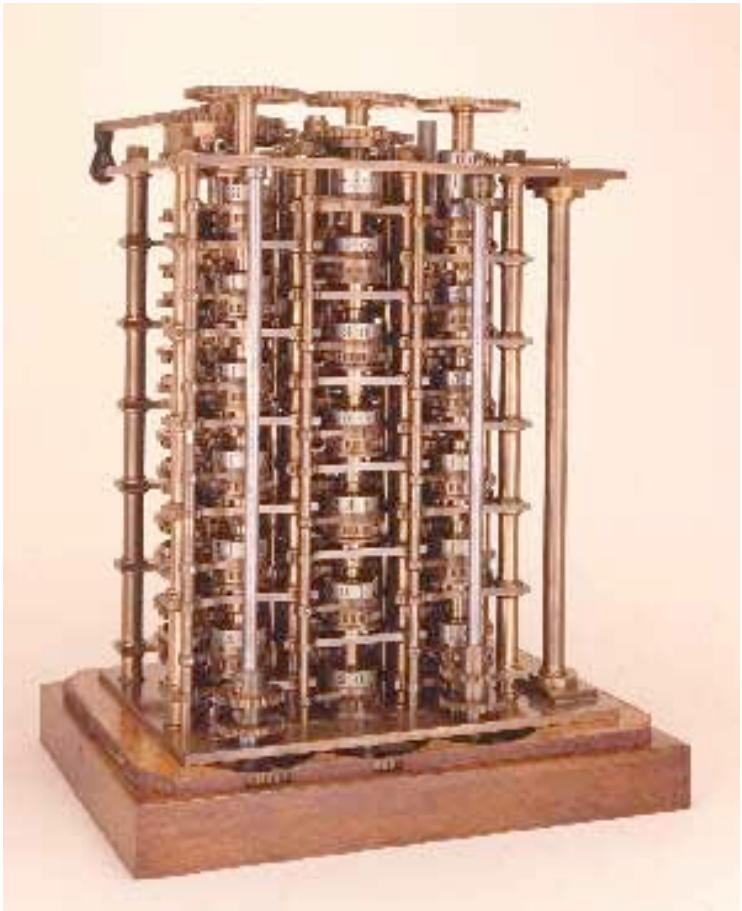
Problème des erreurs (calcul et saisie)

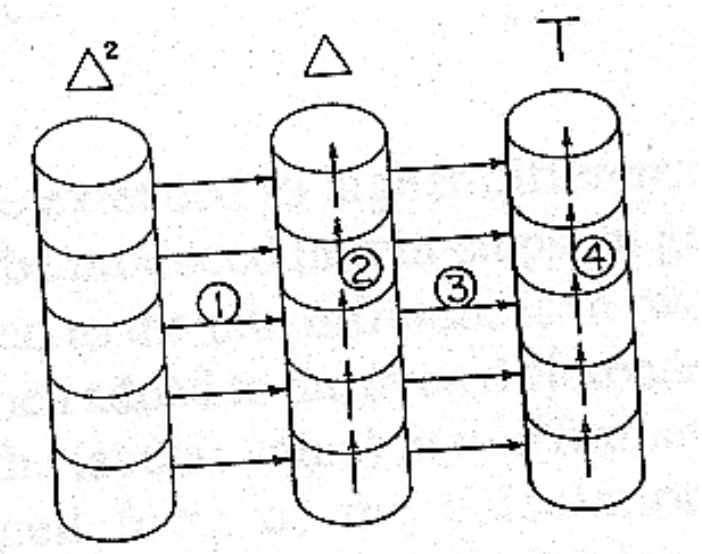
Charles Babbage (1791 - 1871)



1820 : "I wish to God these calculations had been executed by steam"

1832 : La Machine à différences (prototype)





La machine analytique

1833 : "The engine eating its own tail"

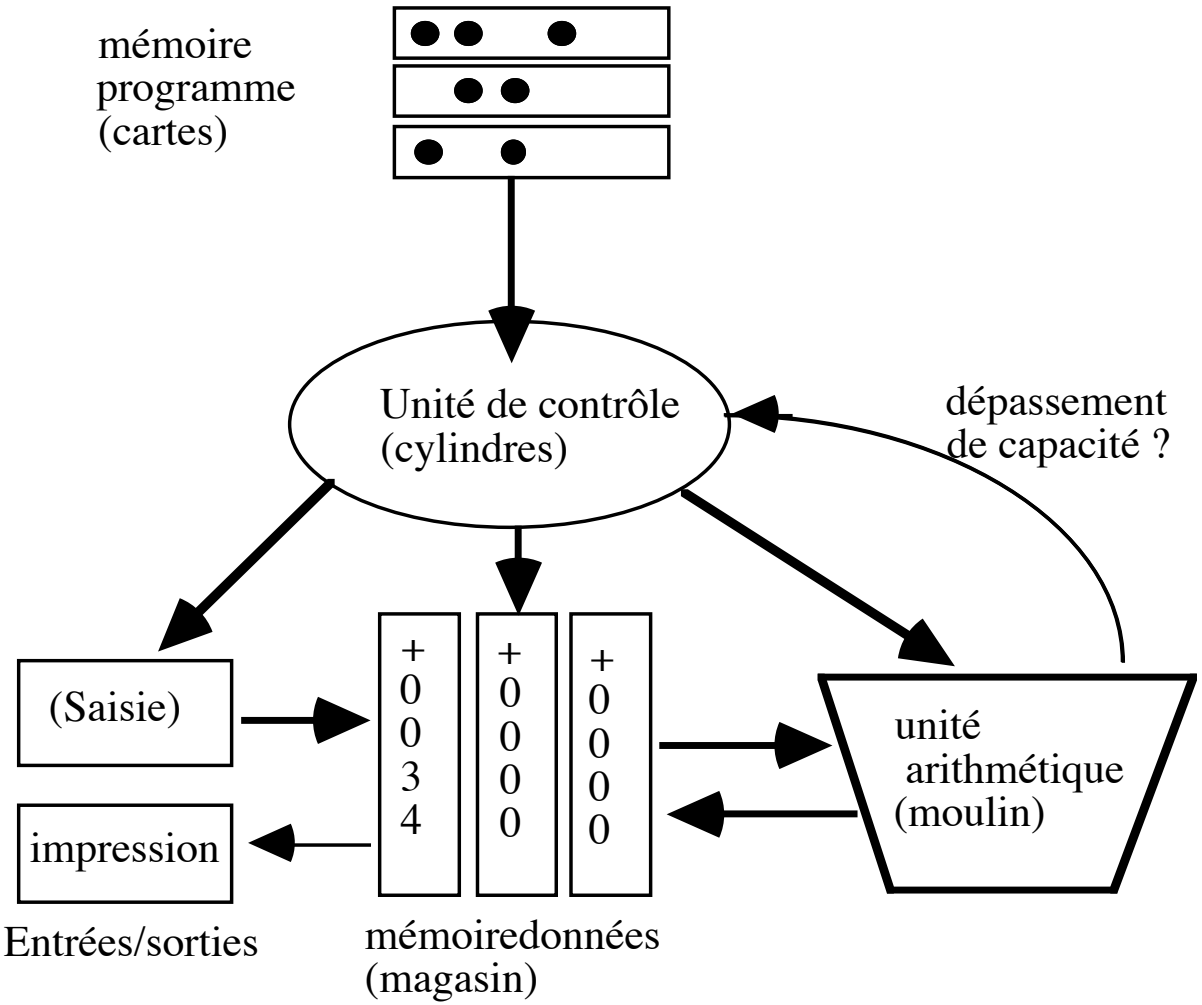
1834 - 36 :

Séparation entre le "store" (magasin des nombres) et le "mill" (moulin, pour le calcul)

La circulation de l'information et la répétition des calculs est controlée par des "barrels" (cylindres à picot)

Contrôle de l'execution (programme) par des cartes perforées (Jacquard)

1838 : Design général finalisé :



1842 : "Sketch of the analytical engine …" par Menabrea, traduit et annoté par Ada Lovelace

Premiers exemples au monde de programmes

Le premier programme :

Résolution d'un système d'équations du premier degré à 2 inconnues :-(

$$\begin{cases} mx + ny = d \\ m'x + n'y = d' \end{cases} \qquad x = \frac{dn' - d'n}{n'm - nm'} \qquad y = \frac{dm' - d'm}{n'm - nm'}$$

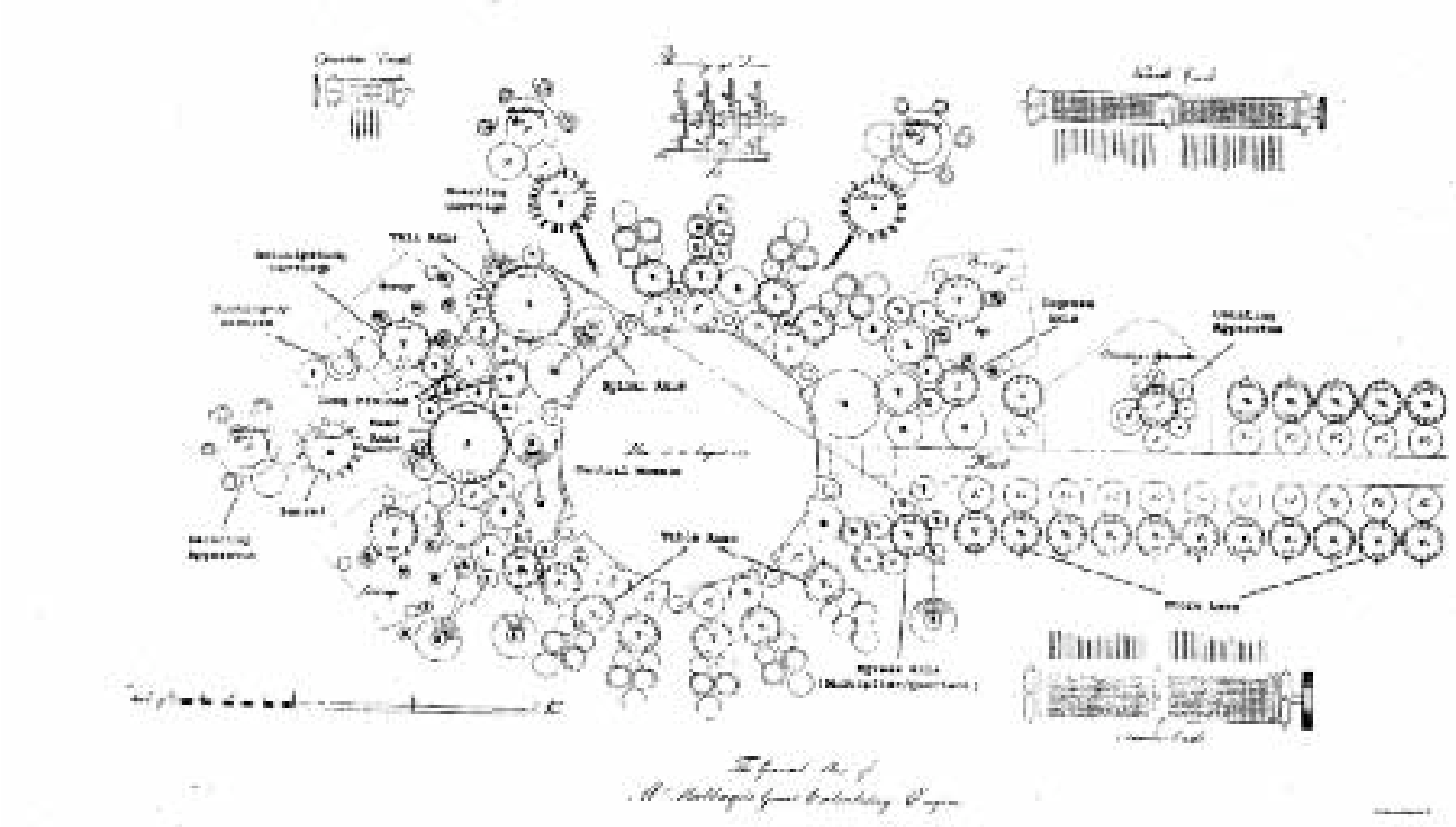
Columns on which are inscribed the primitive data	Number of the operations	Cards of the operations		Variable cards			Statement of results
		No. of the Operation-cards	Nature of each operation	Columns acted on by each operation	Columns that receive the result of each operation	Indication of change of value on any column	
${}^1V_0 = m$	1	1	×	${}^1V_0 \times {}^1V_4 =$	${}^1V_6$	$\left\{ \begin{matrix} {}^1V_0 = {}^1V_0 \\ {}^1V_4 = {}^1V_4 \end{matrix} \right\}$	${}^1V_6 = mn'$
${}^1V_1 = n$	2	"	×	${}^1V_3 \times {}^1V_1 =$	${}^1V_7$	$\left\{ \begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix} \right\}$	${}^1V_7 = m'n$
${}^1V_2 = d$	3	"	×	${}^1V_2 \times {}^1V_4 =$	${}^1V_8$	$\left\{ \begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^1V_4 = {}^0V_4 \end{matrix} \right\}$	${}^1V_8 = dn'$
${}^1V_3 = m'$	4	"	×	${}^1V_5 \times {}^1V_1 =$	${}^1V_9$	$\left\{ \begin{matrix} {}^1V_5 = {}^1V_5 \\ {}^1V_1 = {}^0V_1 \end{matrix} \right\}$	${}^1V_9 = d'n$
${}^1V_4 = n'$	5	"	×	${}^1V_0 \times {}^1V_5 =$	${}^1V_{10}$	$\left\{ \begin{matrix} {}^1V_0 = {}^0V_0 \\ {}^1V_5 = {}^0V_5 \end{matrix} \right\}$	${}^1V_{10} = d'm$
${}^1V_5 = d'$	6	"	×	${}^1V_2 \times {}^1V_3 =$	${}^1V_{11}$	$\left\{ \begin{matrix} {}^1V_2 = {}^0V_2 \\ {}^1V_3 = {}^0V_3 \end{matrix} \right\}$	${}^1V_{11} = dm'$
	7	2	−	${}^1V_6 - {}^1V_7 =$	${}^1V_{12}$	$\left\{ \begin{matrix} {}^1V_6 = {}^0V_6 \\ {}^1V_7 = {}^0V_7 \end{matrix} \right\}$	${}^1V_{12} = mn' - m'n$
	8	"	−	${}^1V_8 - {}^1V_9 =$	${}^1V_{13}$	$\left\{ \begin{matrix} {}^1V_8 = {}^0V_8 \\ {}^1V_9 = {}^0V_9 \end{matrix} \right\}$	${}^1V_{13} = dn' - d'n$
	9	"	−	${}^1V_{10} - {}^1V_{11} =$	${}^1V_{14}$	$\left\{ \begin{matrix} {}^1V_{10} = {}^0V_{10} \\ {}^1V_{11} = {}^0V_{11} \end{matrix} \right\}$	${}^1V_{14} = d'm - dm'$
	10	3	÷	${}^1V_{13} \div {}^1V_{12} =$	${}^1V_{15}$	$\left\{ \begin{matrix} {}^1V_{13} = {}^0V_{13} \\ {}^1V_{12} = {}^1V_{12} \end{matrix} \right\}$	${}^1V_{15} = \frac{dn' - d'n}{mn' - m'n} = x$
	11	"	÷	${}^1V_{14} \div {}^1V_{12} =$	${}^1V_{16}$	$\left\{ \begin{matrix} {}^1V_{14} = {}^0V_{14} \\ {}^1V_{12} = {}^0V_{12} \end{matrix} \right\}$	${}^1V_{16} = \frac{d'm - dm'}{mn' - m'n} = y$
1	2	3	4	5	6	7	8

On peut mieux faire (10 variables au lieu de 16)

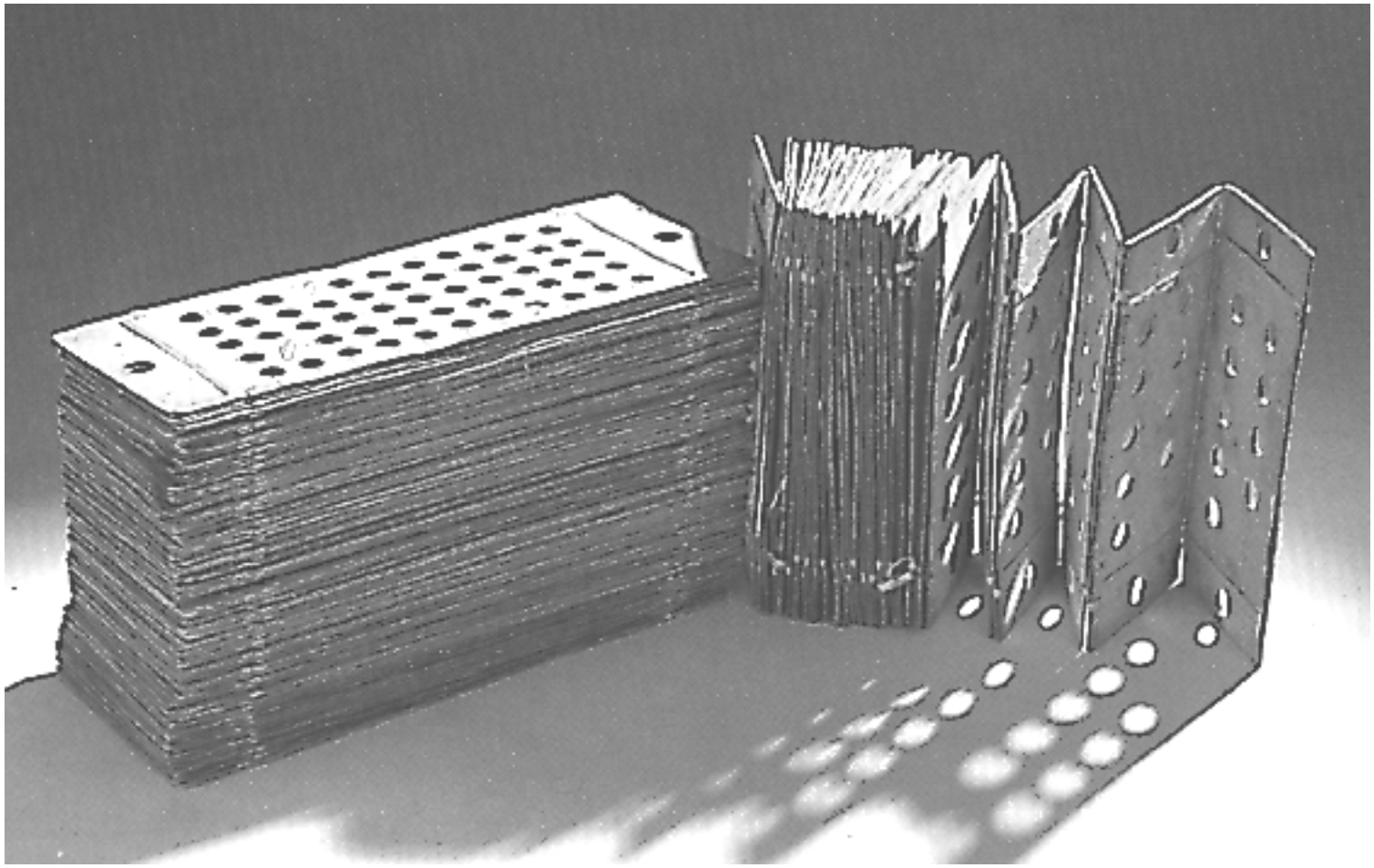
V0 x V4 => V6	(mn’)
V3 x V1 => V7	(m’n)
V6 - V7 => V6	(mn’-m’n)
V2 x V4 => V7	(dn’)
V5 x V1 => V8	(d’n)
V7 - V8 => V7	(dn’-d’n)
V5 x V0 => V8	(d’m)
V2 x V3 => V9	(dm’)
V8 - V9 => V8	(dm’-d’m)
V7 ÷ V6 => V7	(x)
V8 ÷ V6 => V8	(y)

Organisation

A.G. BROMLEY, IEEE Annals of the history of comp. 20(4), 1998, pp. 29-45.



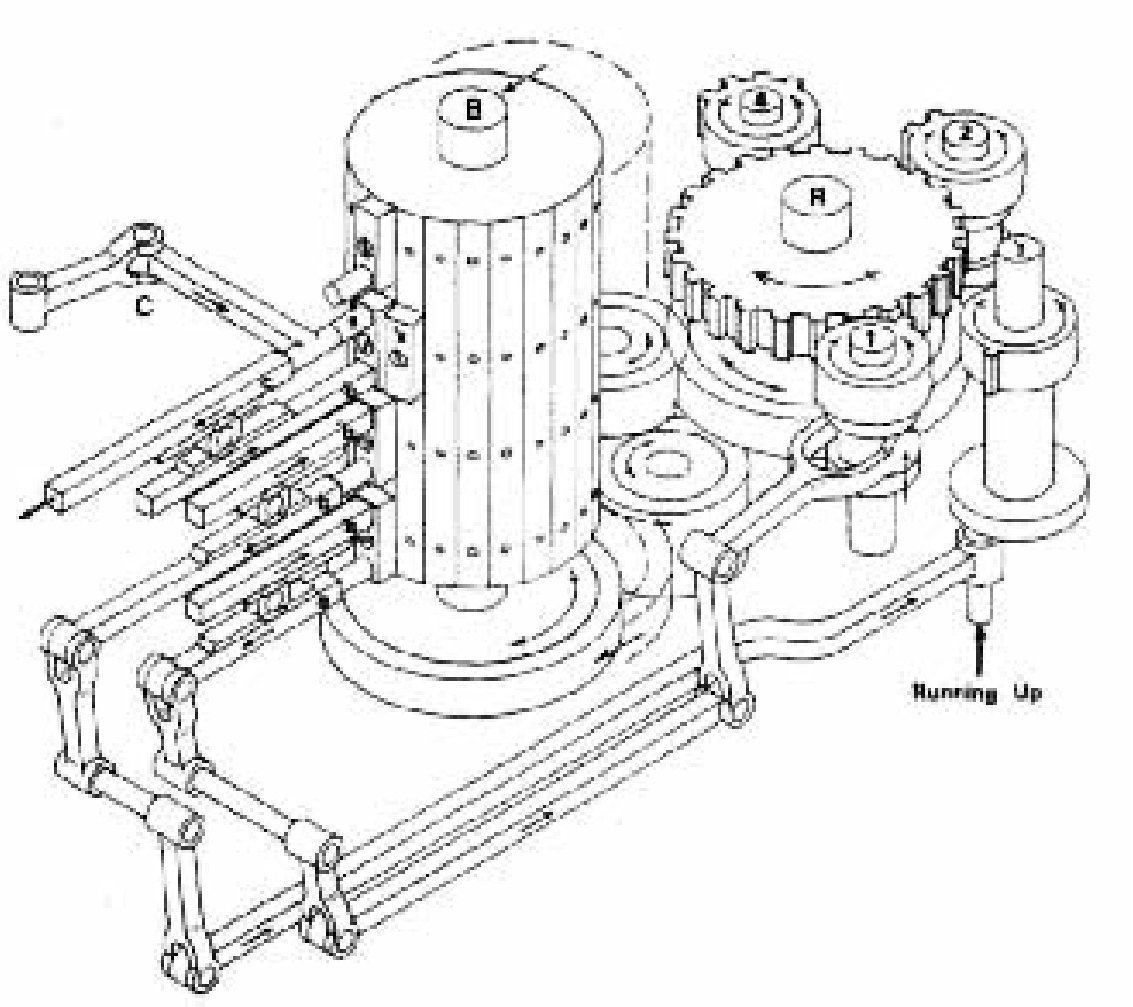
Hauteur 4.60 m Longeur 7.65 m Diamètre moulin 1.85m
200 colonnes (40 chiffres+signe) dans le magasin
200 colonnes de rouages dans le moulin



Cartes de variables et d'opérations

Cycle d'exécution

Toute l'activité de la machine est cadencée par l'unité de contrôle :



1) le prisme porteur de carte pousse des biellettes pour sélectionner le(s) cyclindre(s) concerné(s) par l'opération (les cartes de variables agissent directement)

2) En tournant sur lui-même, le cylindre provoque (par ses picots) le déplacement de biellettes qui dirigent les échanges entres les axes verticaux (du magasin et/ou du moulin)

3) la rotation du cylindre est auto-contrôlée, sauf si le levier de retenue a été levé par le moulin. Le cylindre peut revenir en arrière (traitement itératif).

4) Quand le tour est terminé, la carte suivante est lue

Unité : temps du déplacement pour une valeur sur une roue (0.16 s. env.)

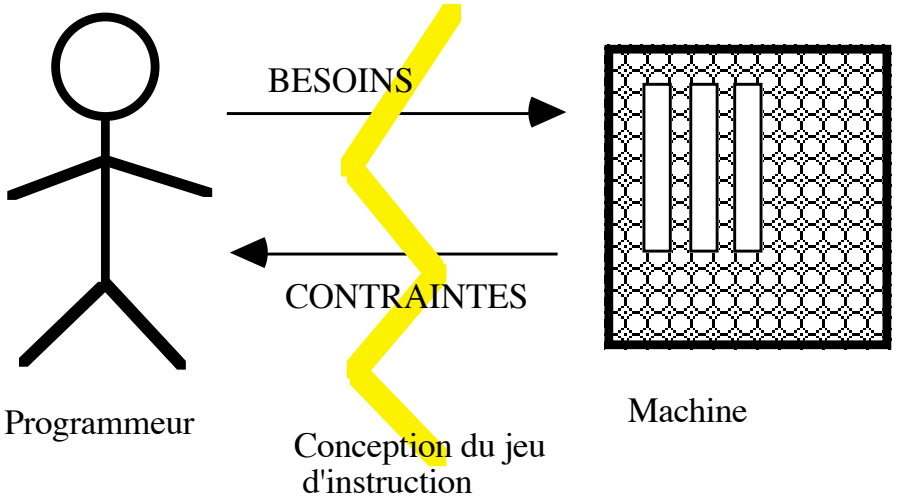
15 u. pour un transfert mémoire/mémoire
20 u. (3 s.) pour une addition avec report retenue
4 min pour une multiplication (au pire)

Jeu d'instructions

L'association cartes/cylindres permet de faire exécuter à la machine toutes sortes d'instructions

- dans les limites du nombre de picots/tour de cylindre (80?)
- à partir des capacités élementaires du moulin

C'est toujours une question essentielle pour le concepteur d'ordinateur :



L'ambition de Babbage était bien de faire un "calculateur universel", mais :

- Sa machine est inadaptée au calcul symbolique, comme le traitement du texte. Elle entrerait aujourd'hui dans la gamme des machines "scientifiques", dédiées au calcul intensif :

```
x <- x0
répéter
   calculer f(x)
   x <- x + dx
jusqu'à plus_soif
```

- Il ne savait pas ce qu'est un nombre calculable (ou non-calculable)

Le jeu d'instructions de Babagge est assez complexe, et nulle part décrit complètement…

Un émulateur (applet Java) écrit par J. Walker :
http://www.fourmilab.ch/babbage/

Une version plus "moderne", avec 2 types de cartes :

opération	opérande 1	opérande 2	destination

```
CPY : V#op1                          -> V#dest
ADD : V#op1 + V#op2                  -> V#dest
SUB : V#op1 - V#op2                  -> V#dest
MUL : V#op1 * V#op2                  -> V#dest
DIV : dividende V#op1 ÷ V#op2-> V#dest
REM : reste V#op1 ÷ V#op2    -> V#dest

EQU : 1( V#op1 = V#op2 )     -> V#dest
INF : 1( V#op1 < V#op2 )     -> V#dest

GET : saisie de V#dest
PUT : impression de V#dest
```

opération	valeur	case

```
SET : valeur                          -> V#case

INC : V#case + valeur                 -> V#case
DEC : V#case - valeur                 -> V#case
OPP : valeur - V#case                 -> V#case

BRA : si V#case = 1 alors avancer
(reculer) de valeur carte(s)

BRN : si V#case = 0 alors avancer
(reculer) de valeur carte(s)

HLT : arrêt de la machine et impression
de valeur
```

Codage des instructions

• Il y a 16 instructions différentes : 2 trous pour les différencier

On peut en ajouter quatre… (ING et MUC, DIC, REC)

• Les nombres sont sur 40 chiffres et un signe : 41 trous pour coder une valeur

Plus pratique ici d'avoir 39 chiffres et un signe = 40 trous

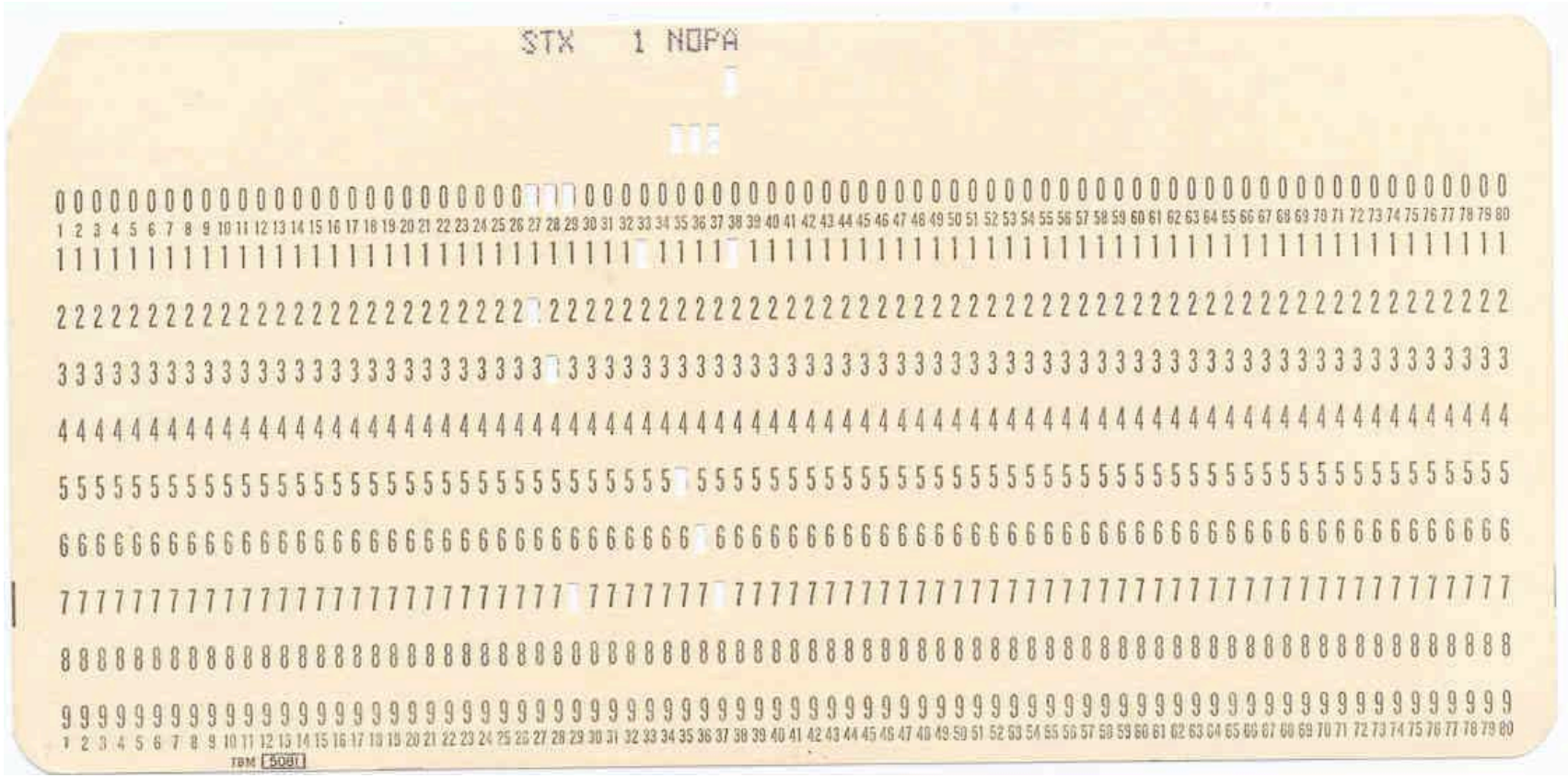
• Si on veut attribuer la même taille de carte aux différents types :

Il faut coder les numéros de cases sur 20 trous

Soit on permet une machine à 10^20 cases, soit on reste avec 1000 cases et on gaspille 76% des trous pour les cartes du premier type

=> carte de 2 + 60 = 62 trous de long et 10 (ou 9) de large

Exemple de codage :



carte IBM 80 colonnes

Un programme type

Calcul de la racine carrée de A par la méthode de Héron :

$$\forall n > 0,\ R_{n+1} = \frac{1}{2}\left(R_n + \frac{A}{R_n} \right) \text{ et } R_0 = A$$

On itère jusqu'à ce que les quarantes chiffres restent les mêmes

Exécution pour A=10 en précision à deux chiffres

R := (10 + 10/10)/2 = 5.50
R := (5.5 + 10/5.5)/2 = 3.66
R := (3.66 + 10/3.66)/2 = 3.20
R := (3.20 + 10/3.20)/2 = 3.16
R := (3.16 + 10/3.16)/2 = 3.16
fini en 5 itérations

Idem en précision à quatre chiffres :

R := (10 + 10/10)/2 = 5.5000
R := (5.5000 + 10/5.5000)/2 = 3.6590
R := (3.6590 + 10/3.6590)/2 = 3.1959
R := (3.1959 + 10/3.1959)/2 = 3.1624
R := (3.1624 + 10/3.1624)/2 = 3.1622
R := (3.1622 + 10/3.1622)/2 = 3.1623
R := (3.1623 + 10/3.1623)/2 = 3.1623
fini en 7 itérations

(voir également polycopié TP VARI)

Attribution des cases :

000	A
001	Rn
002	calculs intermédiaires
003	Rn+1
004	comparaison Rn et Rn+1

Le programme :

```
01    GET//////////   //////////    0…000
02    CPY0…000   //////////    0…001
03    DIV 0…000   0…001   0…002
04    ADD      0…001   0…002   0…002
05    DIC +0……..02   0…003
06    EQU      0…001   0…003   0…004
07    CPY0…003   //////////    0…001
08    BRA      -0……04   0…004
09    PUT//////////   //////////    0…001
10    HLT+0……00   //////////
```

temps d'exécution ???

Reste à perforer les cartes…

Pipeline

Le fonctionnement de la machine analytique peut-être (était ?) optimisé :

Remplacer :

lecture carte instruction 1
exécution 1
lecture carte instruction 2
exécution 2
lecture carte instruction 3
exécution 3
…

Par :

lecture carte instruction 1
exécution 1 // lecture carte instruction 2
exécution 2 // lecture carte instruction 3
exécution 3 // lecture carte instruction 4
…

Mieux encore :

lecture 1
rech. opérandes 1 // lecture 2
calcul 1 // rech. op 2 // lecture 3
stockage résultat 1 // calcul 2 // rech. op. 3 // lecture 4
stockage 2 // calcul 3 // rech. op. 4 // lecture 5
stockage 3 // calcul 4 // rech. op. 4 // lecture 6
stockage 4 // calcul 5 // rech. op. 4 // lecture 7
…

Technique du "pipeline" qui est utilisée systématiquement dans les processeurs actuels

Limité par les dépendances entre les opérandes d'instructions successives (données)

Par les branchements conditionnels (traitement)

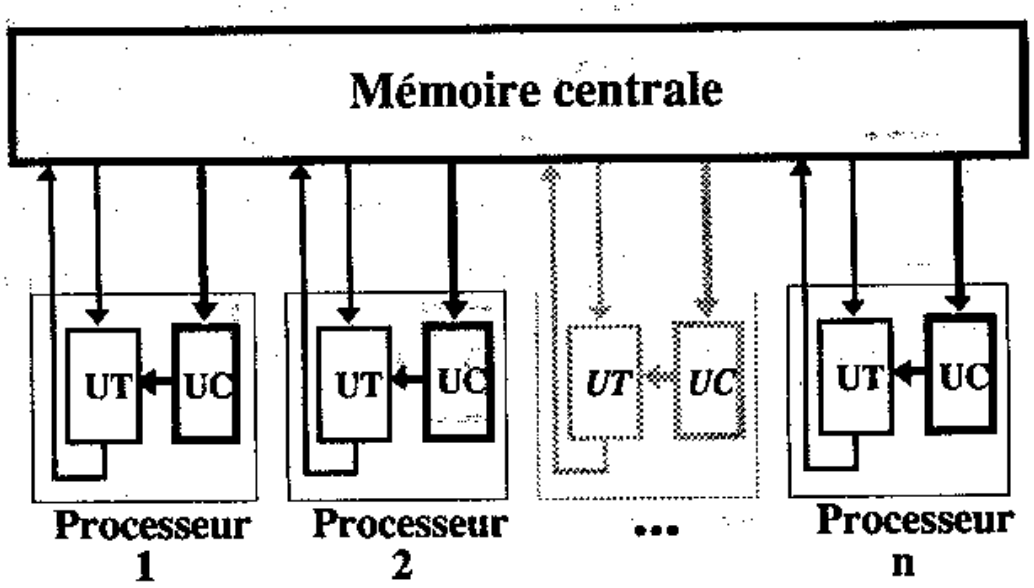
Parallélisme

Les chiffres des opérandes étaient traités quasi-simultanément grâce à un mécanisme de retenue anticipée

La présence de plusieurs cylindres (3 sur le plan, 7 prévus) permettait de traiter plusieurs opérations simultanément

Classification de Flynn :

SISD : Single Instruction - Single Data
SIMD : Single Instruction - Multiple Data ("vectoriel")
MISD : Multiple Instruction - Single Data (pipeline)
MIMD : Multiple Instruction Multiple Data (multiproc.)



Solution retenue actuellement :

SIMD + MISD = VLIW (very long instruction word)
Pipeline des instructions + multiplication des unités de calculs

Le futur :

Interconnexion de VLIW pour en faire une machine MIMD ?