- VARI -

Architecture des ordinateurs

Pierre Cubaud
cubaud@cnam.fr

Octobre 2000

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"

Bibliographie

• 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, …) :

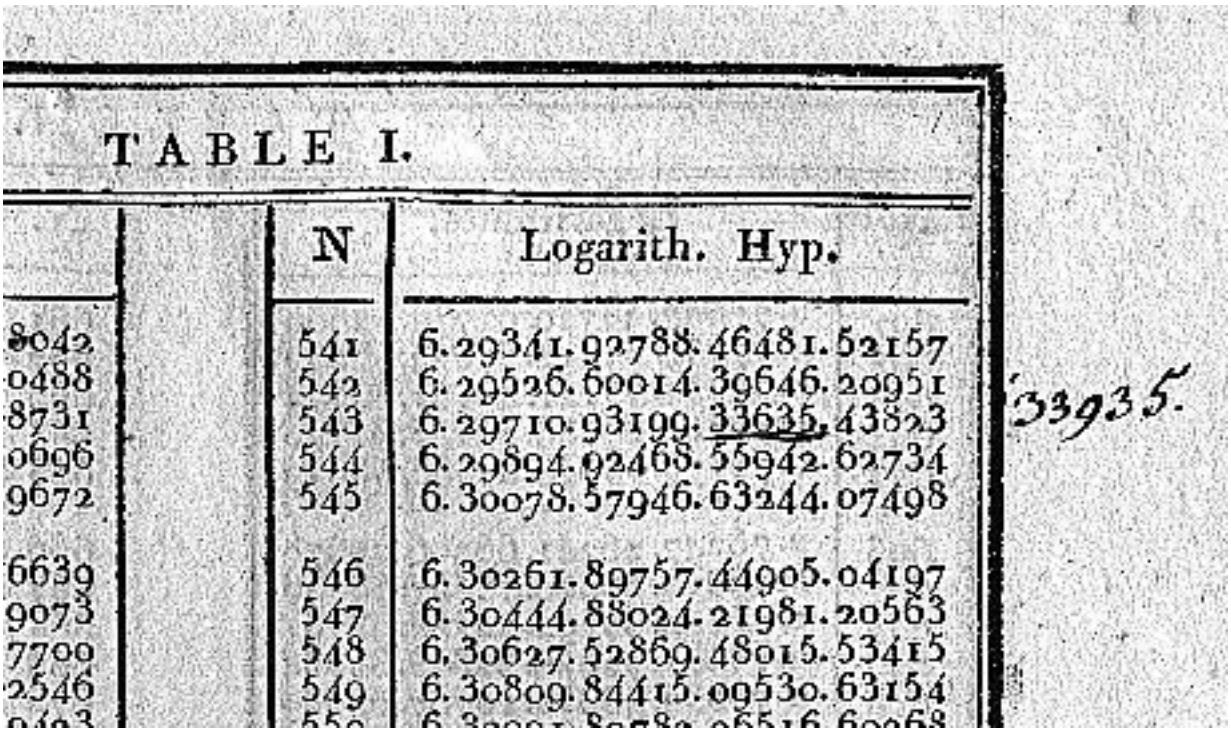


TABLE I.

	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 :

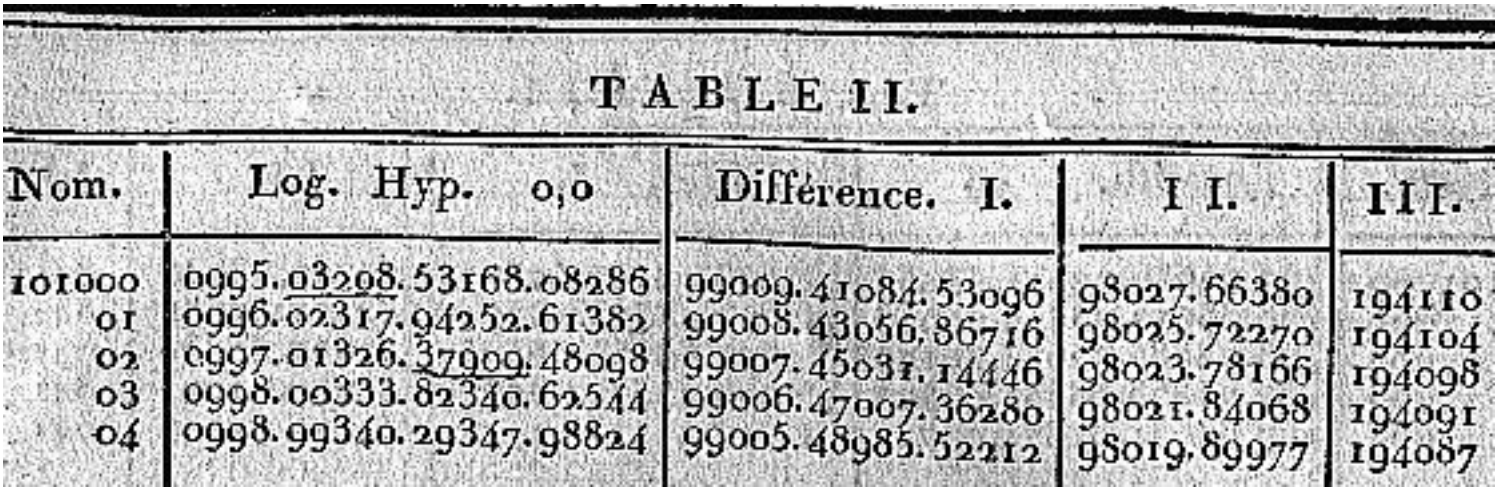


TABLE II.

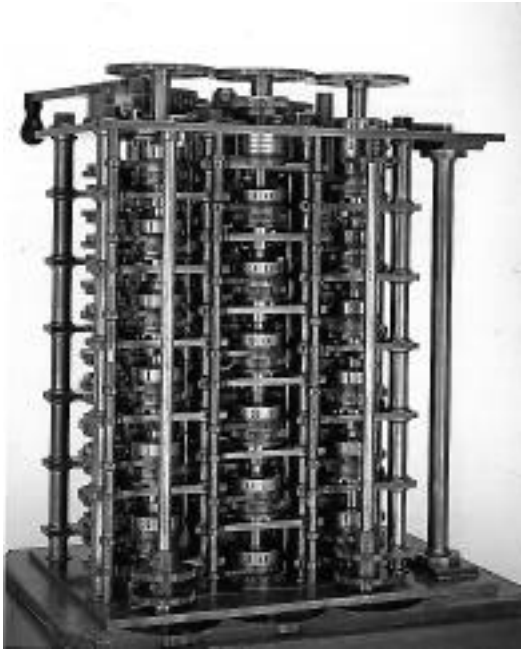
Nom.	Log. Hyp. 0,0	Différence. I.	I I.	I I I.
101000	0995.03208.53168.08286	99009.41084.53096	98027.66380	194110
01	0996.02317.94252.61382	99008.43056.86716	98025.72270	194104
02	0997.01326.37909.48098	99007.45031.14446	98023.78166	194098
03	0998.00333.82340.62544	99006.47007.36280	98021.84068	194091
04	0998.99340.29347.98824	99005.48985.52212	98019.89977	194087

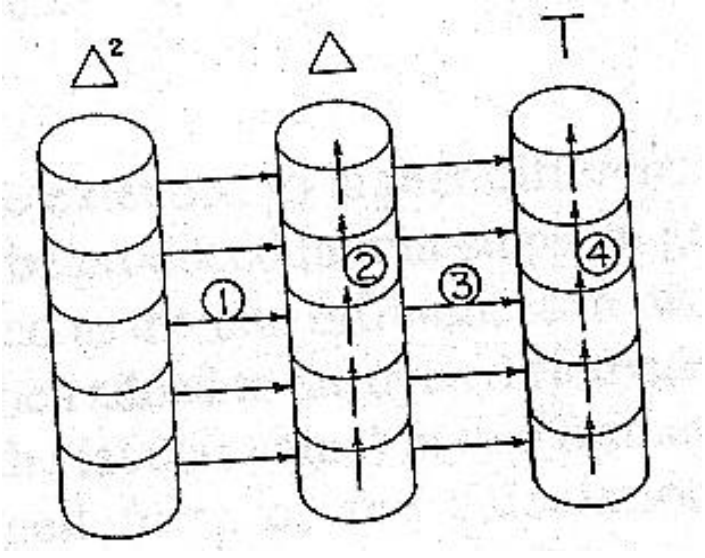
Problème des erreurs (calcul et saisie)

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"

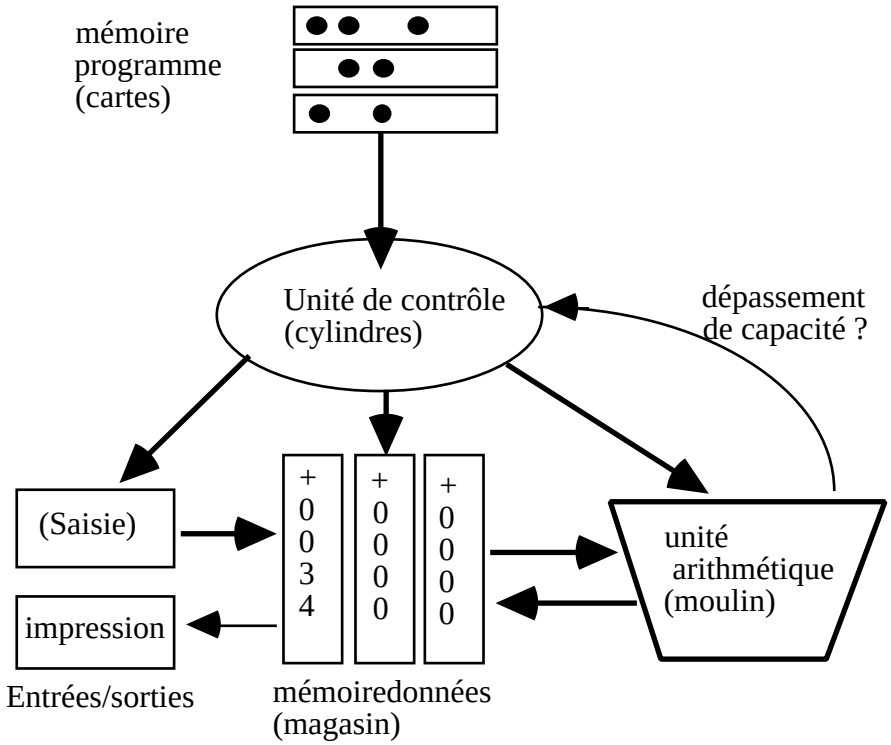
1834 - 36 :

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	×	${}^1V_0 \times {}^1V_4 =$	${}^1V_6$	$\begin{cases} {}^1V_0 = {}^1V_0 \\ {}^1V_4 = {}^1V_4 \end{cases}$	${}^1V_6 = mn'$
${}^1V_1 = n$	2	"	×	${}^1V_3 \times {}^1V_1 =$	${}^1V_7$	$\begin{cases} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{cases}$	${}^1V_7 = m'n$
${}^1V_2 = d$	3	"	×	${}^1V_2 \times {}^1V_4 =$	${}^1V_8$	$\begin{cases} {}^1V_2 = {}^1V_2 \\ {}^1V_4 = {}^0V_4 \end{cases}$	${}^1V_8 = dn'$
${}^1V_3 = m'$	4	"	×	${}^1V_5 \times {}^1V_1 =$	${}^1V_9$	$\begin{cases} {}^1V_5 = {}^1V_5 \\ {}^1V_1 = {}^0V_1 \end{cases}$	${}^1V_9 = d'n$
${}^1V_4 = n'$	5	"	×	${}^1V_0 \times {}^1V_5 =$	${}^1V_{10}$	$\begin{cases} {}^1V_0 = {}^0V_0 \\ {}^1V_5 = {}^0V_5 \end{cases}$	${}^1V_{10} = d'm$
${}^1V_5 = d'$	6	"	×	${}^1V_2 \times {}^1V_3 =$	${}^1V_{11}$	$\begin{cases} {}^1V_2 = {}^0V_2 \\ {}^1V_3 = {}^0V_3 \end{cases}$	${}^1V_{11} = dm'$
	7	2	−	${}^1V_6 - {}^1V_7 =$	${}^1V_{12}$	$\begin{cases} {}^1V_6 = {}^0V_6 \\ {}^1V_7 = {}^0V_7 \end{cases}$	${}^1V_{12} = mn' - m'n$
	8	"	−	${}^1V_8 - {}^1V_9 =$	${}^1V_{13}$	$\begin{cases} {}^1V_8 = {}^0V_8 \\ {}^1V_9 = {}^0V_9 \end{cases}$	${}^1V_{13} = dn' - d'n$
	9	"	−	${}^1V_{10} - {}^1V_{11} =$	${}^1V_{14}$	$\begin{cases} {}^1V_{10} = {}^0V_{10} \\ {}^1V_{11} = {}^0V_{11} \end{cases}$	${}^1V_{14} = d'm - dm'$
	10	3	÷	${}^1V_{13} \div {}^1V_{12} =$	${}^1V_{15}$	$\begin{cases} {}^1V_{13} = {}^0V_{13} \\ {}^1V_{12} = {}^1V_{12} \end{cases}$	${}^1V_{15} = \frac{dn' - d'n}{mn' - m'n} = x$
	11	"	÷	${}^1V_{14} \div {}^1V_{12} =$	${}^1V_{16}$	$\begin{cases} {}^1V_{14} = {}^0V_{14} \\ {}^1V_{12} = {}^0V_{12} \end{cases}$	${}^1V_{16} = \frac{d'm - dm'}{mn' - m'n} = y$
1	2	3	4	5	6	7	8

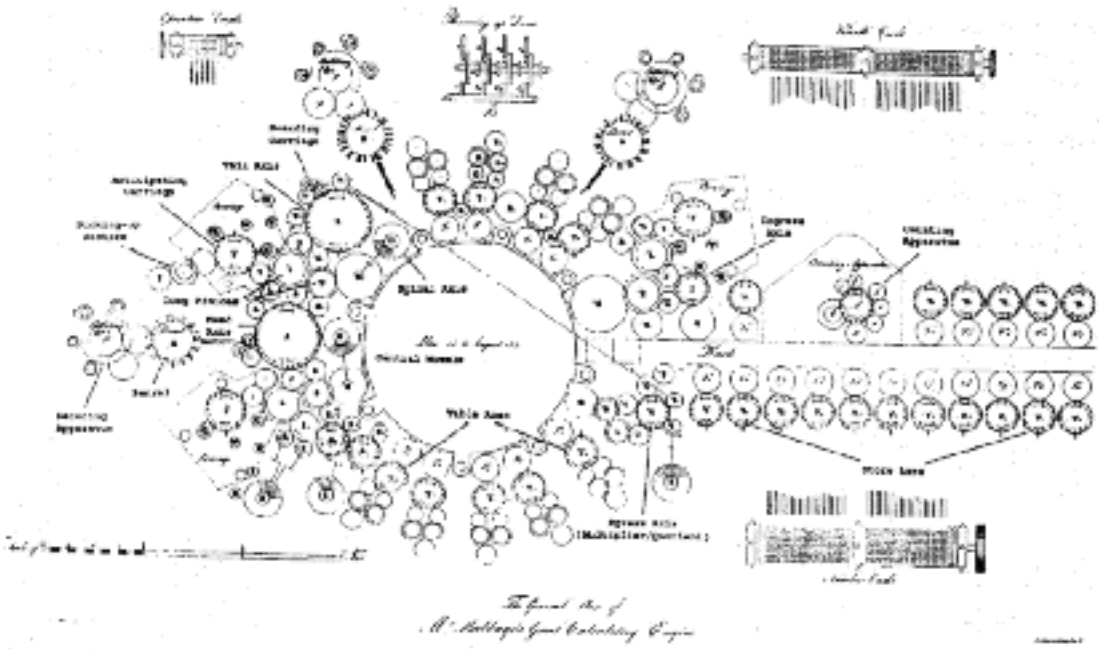
On peut mieux faire (9 variables, 10 op.)

V0 x V4 = V6
V3 x V1 = V7
V6 - V7 = V6
V2 x V4 = V7
V5 x V1 = V8
V7 - V8 = V7

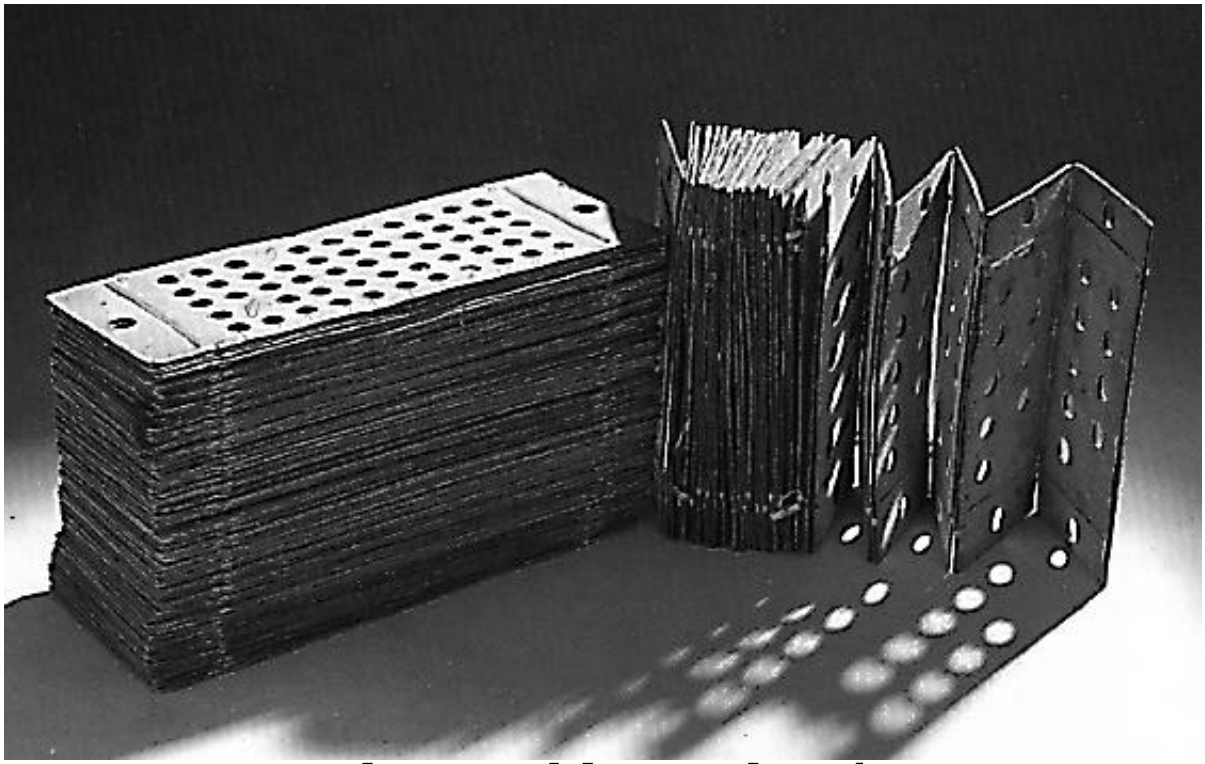
V0 x V5 = V8
V2 x V3 = V9
V8 - V9 = V9
V7 ÷ V6 = V7 (x)
V8 ÷ V6 = V8 (y)

Organisation

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



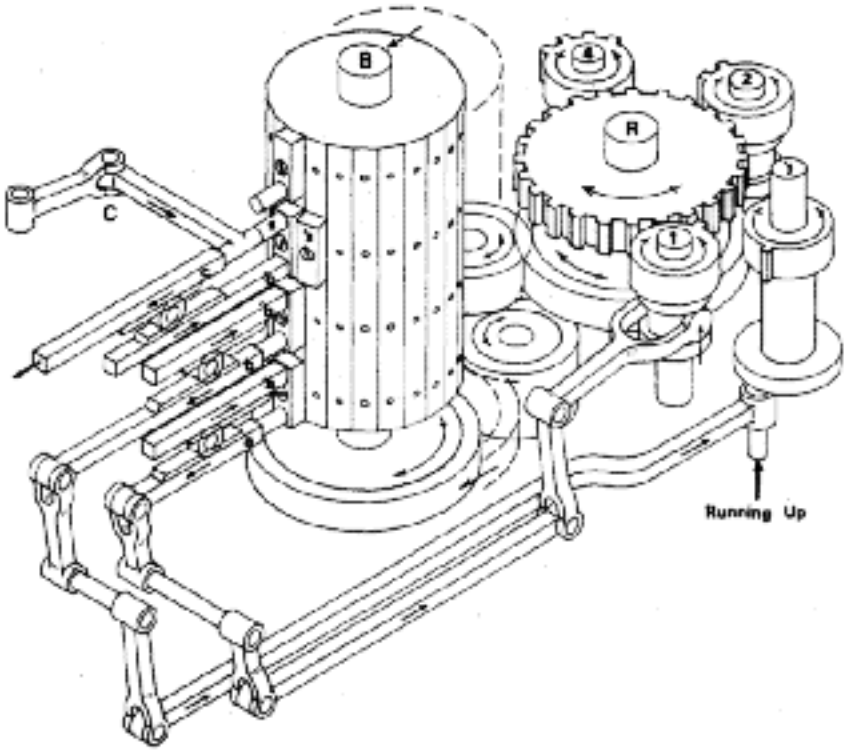
Hauteur 4.60 m Longeur 7.65 m Diamètre moulin 1.85m
200 colonnes (40 chiffres+signe) dans le magasin
200 colonnes de rouages dans le moulin



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.)

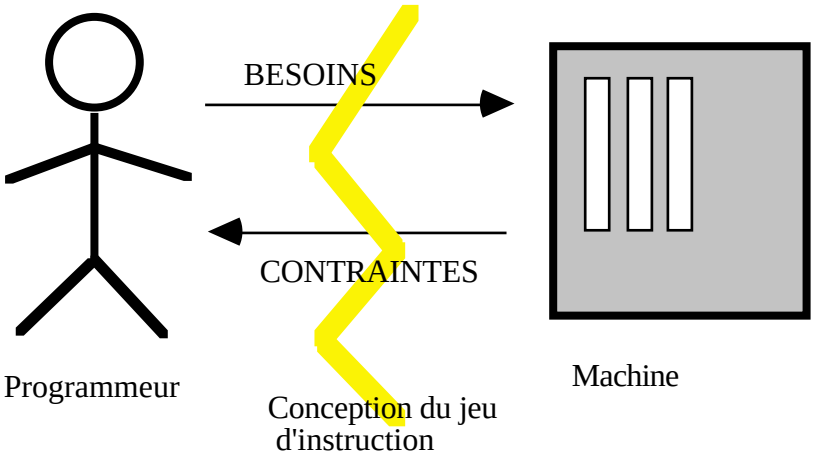
15 u. pour un transfert mémoire/mémoire
20 u. (3 s.) pour une addition avec report retenue
4 min pour une multiplication (au pire)

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

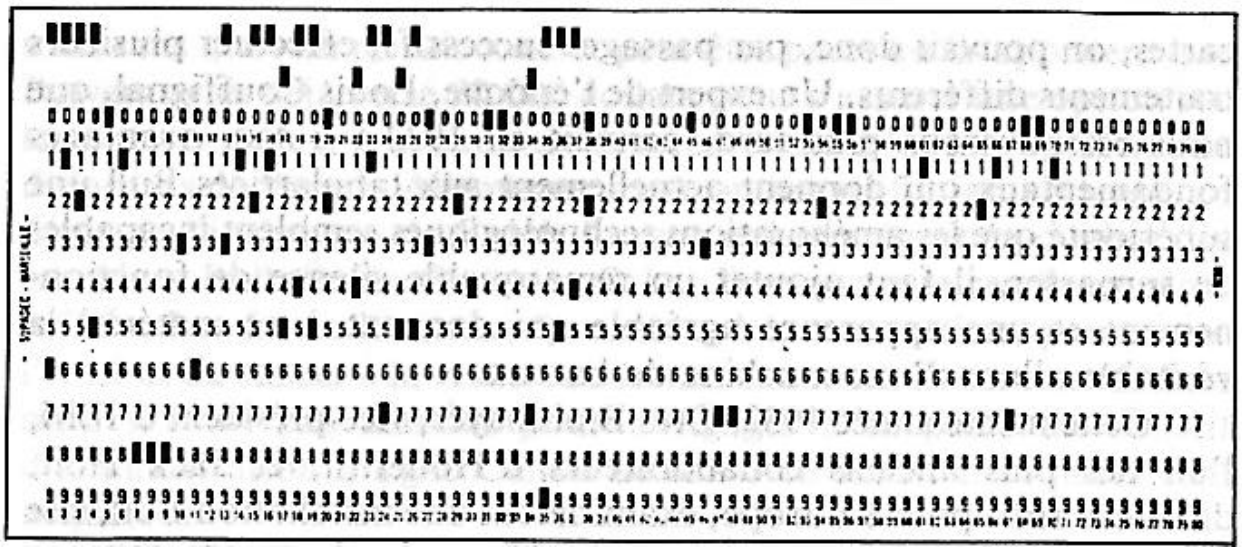
• Si on veut attribuer la même taille de carte aux différents types :

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 :



Un programme type

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

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

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    CPY     0…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    CPY     0…003        //////////   0…001
08    BRA     -0……04                    0…004
09    PUT     //////////   //////////   0…001
10    HLT     +0……00                    //////////
```

Reste à perforer les cartes…

Temps d'exécution ???

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)

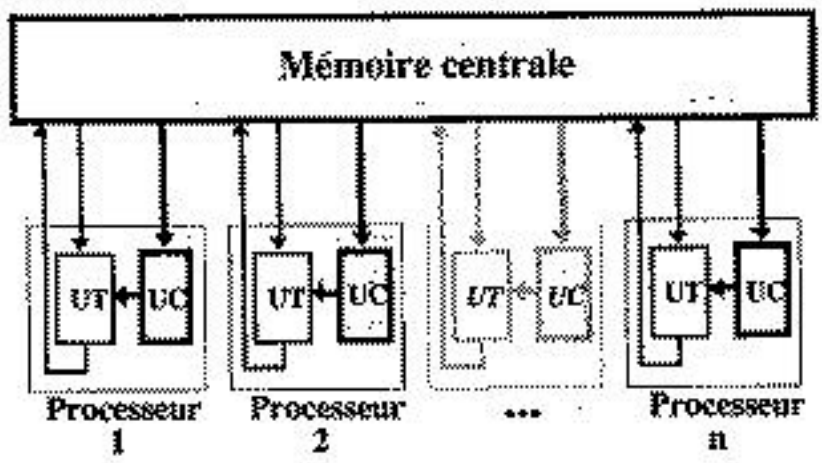
Parallélisme

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 ?

Acte II : 1945

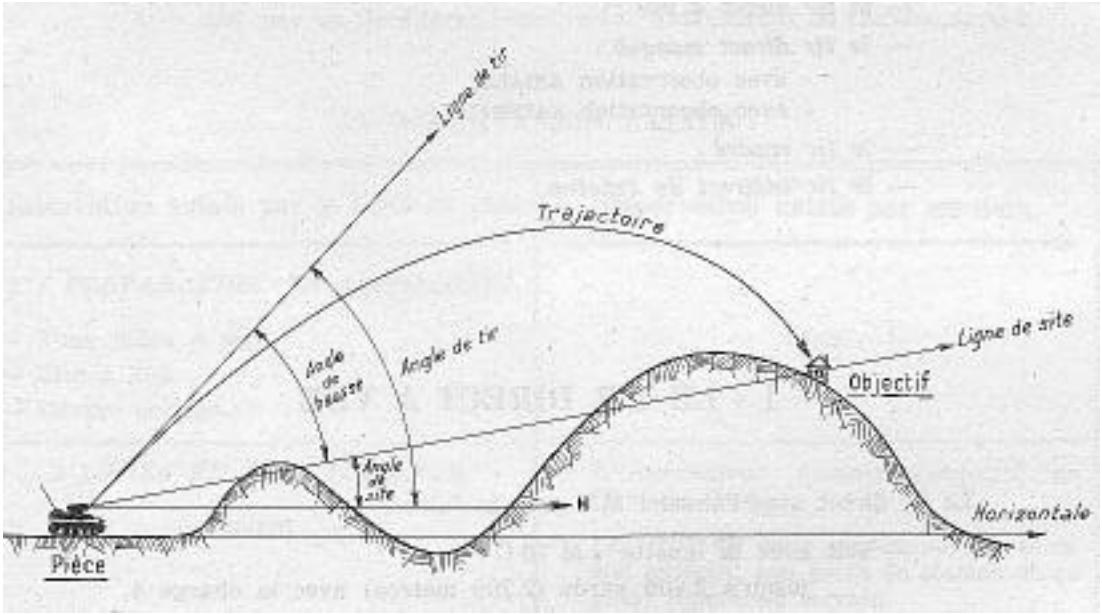


TABLE DE TIR DE L'OBUSIER DE 75 millimètres

DISTANCE		CHARGE 1 (1)		CHARGE 2 (2)		CHARGE 3 (3)		CHARGE 4 (4)	
Mètres	Yards	Hausse millièmes (a)	Fourchette millièmes (b)	Hausse millièmes (c)	Fourchette millièmes (d)	Hausse millièmes (e)	Fourchette millièmes (f)	Hausse millièmes (g)	Fourchette millièmes (h)
0	0	— 5	0	— 4	0	— 3	0	0,6	0
91	100	+ 5	4	3	2	2	1	2	1
183	200	15	4	10	2	7	1	6	1
274	300	25	6	17	2	12	1	9	1
366	400	35	6	25	4	18	1	12	1
457	500	45	6	32	4	23	1	15	1
548	600	56	8	40	4	29	1	19	1
640	700	67	8	48	4	35	1	22	1
731	800	78	8	56	6	40	2	25	1
823	900	89	8	64	6	46	2	29	1
914	1.000	100	10	72	6	52	2	33	1
1.005	1.100	111	10	84	6	58	2	37	1
1.097	1.200	122	10	89	6	64	2	40	1
1.188	1.300	134	10	97	8	69	2	44	1
1.280	1.400	146	12	106	8	76	2	48	1
1.371	1.500	158	12	114	8	82	2	52	1
1.462	1.600	170	12	132	8	88	2	57	2
1.554	1.700	182	14	138	8	94	4	61	2

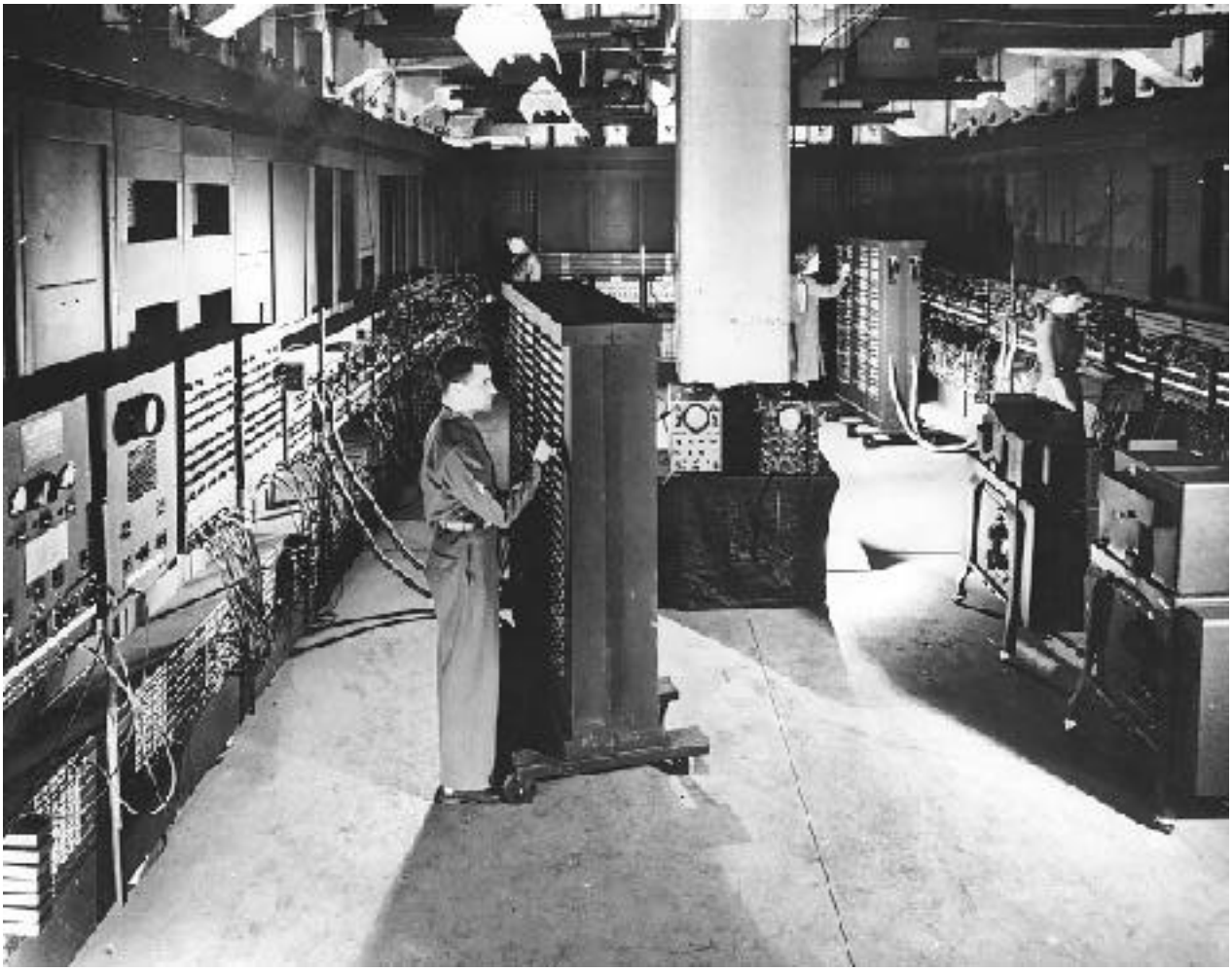
360° = 6400 millièmes

Table de tir : 6 mois / homme de calculs

Août 44 : 6 demandes de tables /j au BRL (Goldstine)

L'arrivée de l'électronique

Electronic Numerical Integrator And Computer



Cadencé par une horloge à 0.1 MHz

Addition : 0.2 ms
Multiplication : 3 ms
Division : 30 ms

20 cases mémoires de 10 chiffres

18000 tubes
70000 résistances, 10000 condensateurs, 6000 interupteurs

Consommation de 140 KW

35 m de long, 3 m de haut, 12 cm de profondeur, 30 tonnes

EDVAC : von Neuman (et al.)

Juin 1945 : "First draft of a report on the EDVAC" par John von Neuman

1946 : "Preliminary discussion of the logical design of an electronic computing instrument" par Burks, Goldstine et von Neuman

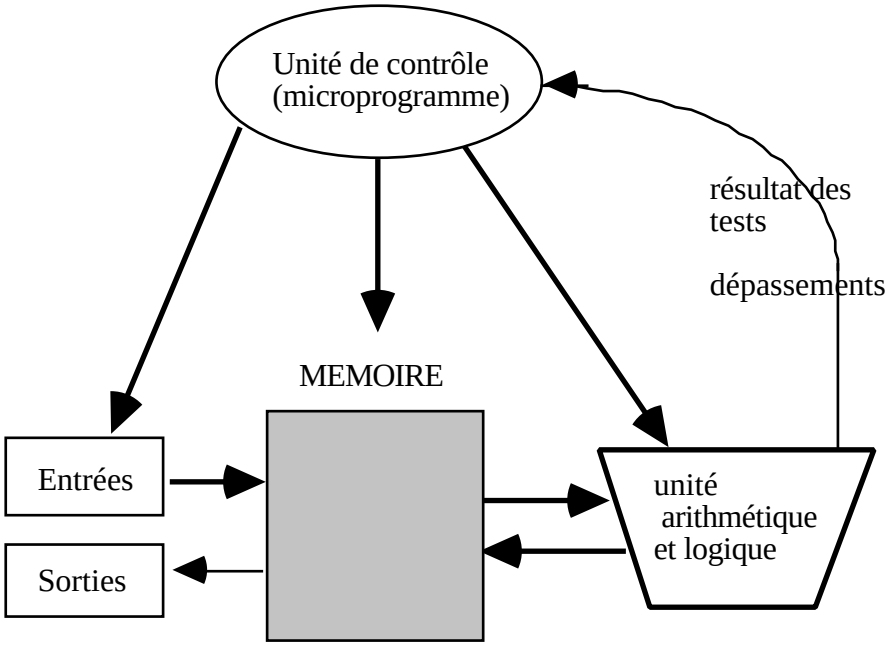
Une machine binaire (calculs et mémorisation)

4000 cases de 40 bits (2^40 = 10^12 bits)

Horloge de base 1MHz

Le programme est conservé dans la même mémoire que les données : il devient lui aussi susceptible de modifications

=> définition de l'ordinateur : calculateur automatique à programme enregistré



II.1 La question de la mémoire

"in the solution of partial differential equations, the storage requirements are likely to be quite extensive"

De l'invention de l'ordinateur à nos jours, il y a toujours eu "pénurie" de mémoire.

La mémoire nécessaire à un programme moyen augmente de 1.5 à 2 par an => 1 bit d'adresse par an. (Hennesy/Patterson)

Technologie idéale :

- Duplicable à faible coût
- Fiable (taux d'erreur /bit)

EQUILIBRE DES DEBITS : Temps d'accès (écriture/lecture) compatible au temps effectué par les calculs du processeur.

Régle d'Amdahl/Case : un système est equilibré, si pour chaque MIPS, il dispose de 1 Mo de mémoire et de 1 Mb/s de débit E/S

Technologies employées

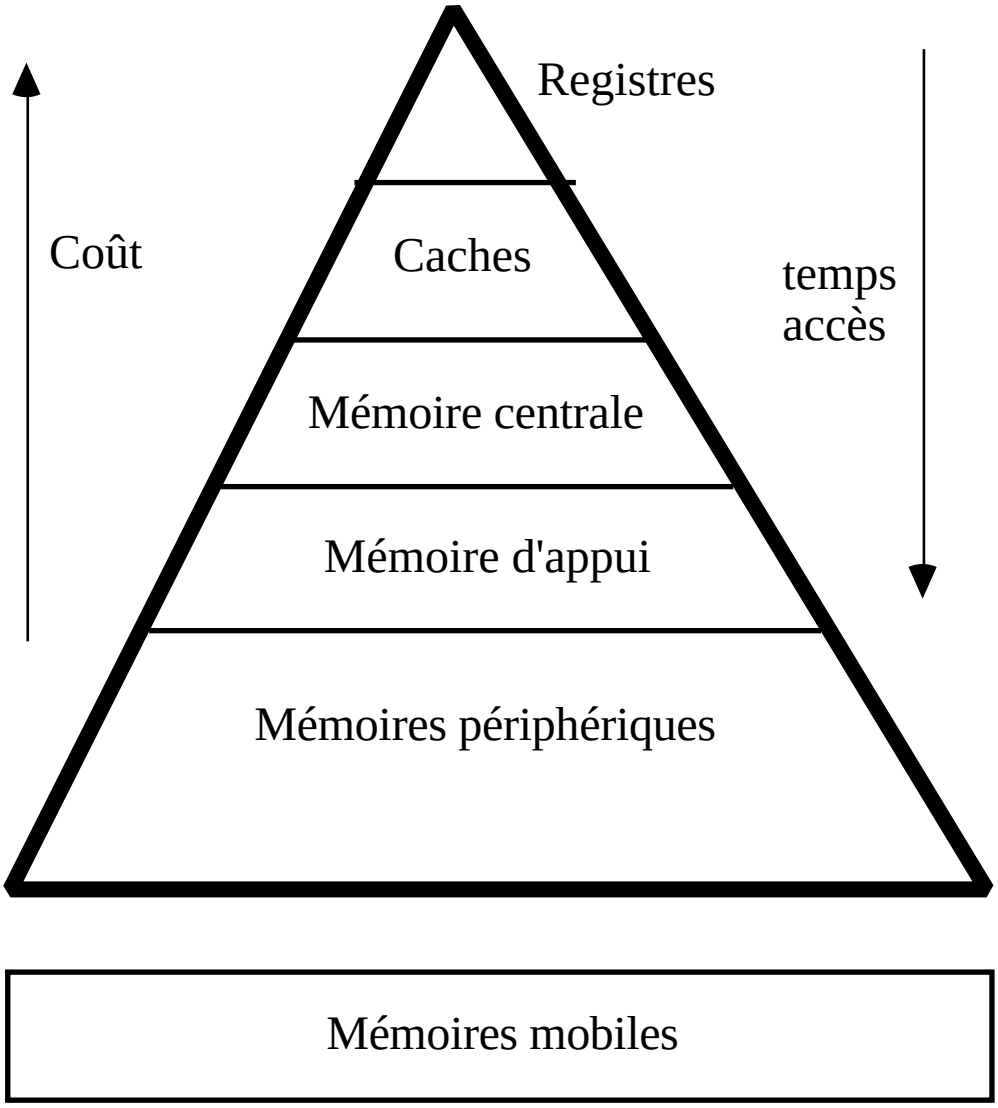
Duplication coûteuse : le bistable ("flip-flop")
- relais électromécaniques
- tubes à vide
- transistor à jonction

Duplication "massive" :
- Cylindre électrostatique (effet capacitif)
- Rémanence écran
- Lignes à retard (ultrason dans mercure)
- Tores de ferrite
- Bande magnétique
- Disque magnétique

1970 : Intel produit le premier circuit intégré de mémoire (1024 bits) sur 0.25 mm2 equivalent à 2 m2 de tores

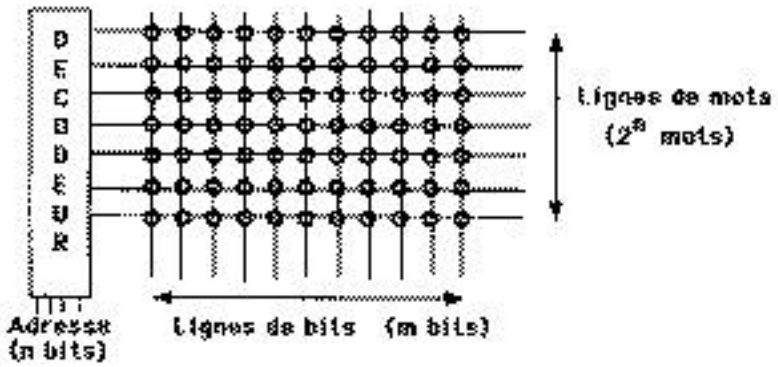
Hiérarchie de mémoires

Compromis toujours nécéssaire entre les besoins des programmes et les possibilités technologiques (à coût fixé)



Taxonomie

- Accès direct



  - RAM : random access memory

    RAM statique
    RAM dynamique : doit être rafraichie régulièrement (- cher)

  - ROM : read-only memory

  - REPROM : reprogrammable ROM : EPROM (uv) EEPROM

- Accès séquentiel

  - cartes perforées (ROM)
  - bande magnétique (RAM)

- Accès cyclique

  - tambour éléctrostatique
  - tambour magnétique

- Accès séquentiel/cyclique

  - disque magnétique
  - disque optique (ROM) et magnéto-optique (REPROM)

- Accès par le contenu (mémoire associative)

  - utilisé pour les mémoires caches (+ complexe => + cher)

Quelques ordres de grandeurs

Capacité mémoire : en bits

1 octet = 8 bits = 1 byte (pas toujours)
1 o = 1 caractère (table d'encodage ISOLatin1)
3 o = un pixel sur un écran "million de couleur"
4 o = 1 nombre réel dans la norme IEEE-754

2 Ko (kilo) = 10^3 o = une page de texte

1Mo (mega) = 10^6 o = Un roman en mode texte
20 Mo = Un roman en fac-similé = Un logiciel
500 Mo = Texte de l'Encyclopedia Universalis (CDROM)

1 Go (giga) = 10^9 o
= Une bibliothèque personnelle en mode texte
10 Go = Un film compressé (DVD)
600 Go = Une librairie en fac-similé (300 Kvol.)

1 To (tera) = 10^12 o = Une cinémathèque personnelle
20 To = plus grand assemblage de disques en 1996 (LNB)
= la "library of congress" en mode texte (50 Mvol)

1 Po (peta) = 10^15 o
= Une bibliothèque nationale en mode image
15 Po = production mondiale de disques en 1995
200 Po = production mondiale de bandes magnétiques en 1995

Temps d'accès (≠ tps de cycle) : en secondes

1 ms (milli) = 10^-3 s
20 ms : tps accès maximum pour un relais
1 µs (micro) = 10^-6 s = tps accès maximum pour un tore
1 ns (nano) = 10^-9s
100 ns = temps d'accès DRAM
5 ns = temps d'accès bascule élementaire
1 ps (pico) = 10^-12s
temps pour qu'un signal lumineux ds le vide parcoure 3 cm

II.2 Le calcul binaire

Connu depuis longtemps comme curiosité

Yi-King chinois, popularisé en Europe par Leibnitz
cf IFRAH Hist. des chiffres, Laffont

Conversion base 10 <=> base 2

Exprimer 67 en base 2 :

On effectue des divisions entières successives en mémorisant le reste :

67	divisé par 2 = 33	reste	1
33	divisé par 2 = 16	reste	1
16	divisé par 2 = 8	reste	0
8	divisé par 2 = 4	reste	0
4	divisé par 2 = 2	reste	0
2	divisé par 2 = 1	reste	0
1	divisé par 2 = 0	reste	1

Lecture des reste de bas en haut pour trouver 1000011

Inversement 1000011 vaut en base 10 :
$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1$
$= 2^6 + 2^1 + 1 = 64 + 2 + 1$
$= 67$

Addition binaire 107 + 18 = 125

107 = 1101011
18 = 0010010
---
125 = 1111101

Multiplication binaire 107 * 18 = 1926

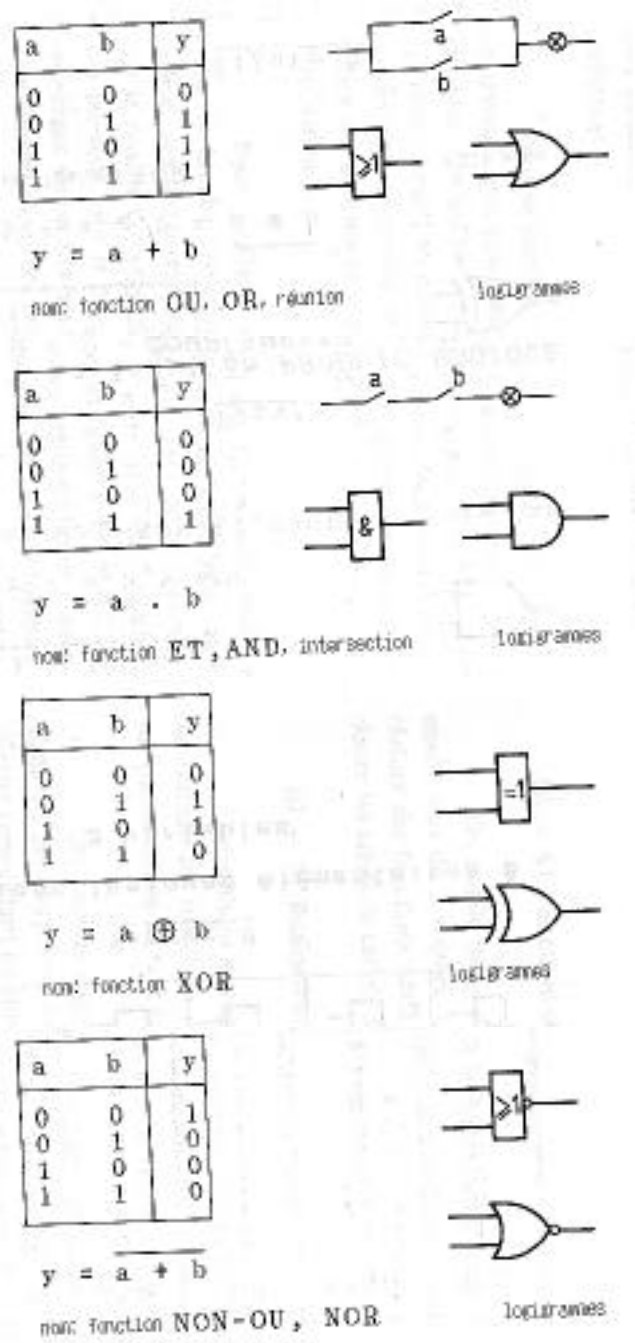
••••87654321 rang
••••1101011• 1er multiplicande décalé à 2
1101011••••• 2ème multiplicande décalé à 5

11110000110 addition des deux donne le total

Fonctions logiques

4 fonctions logiques pour une variable

16 fonctions pour 2 variables



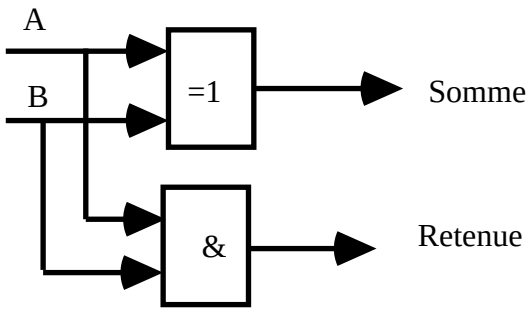
Lois de Morgan : non (A et B) = non(A) ou non(B)
non(A ou B) = non(A) et non(B)

Circuit additionneur

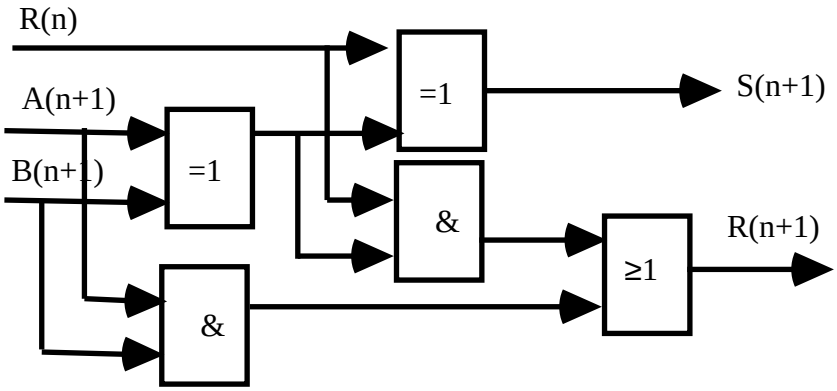
Table de vérité de la somme S = A + B

AB :	00	01	10	11
Somme :	0	1	1	0
Retenue :	0	0	0	1

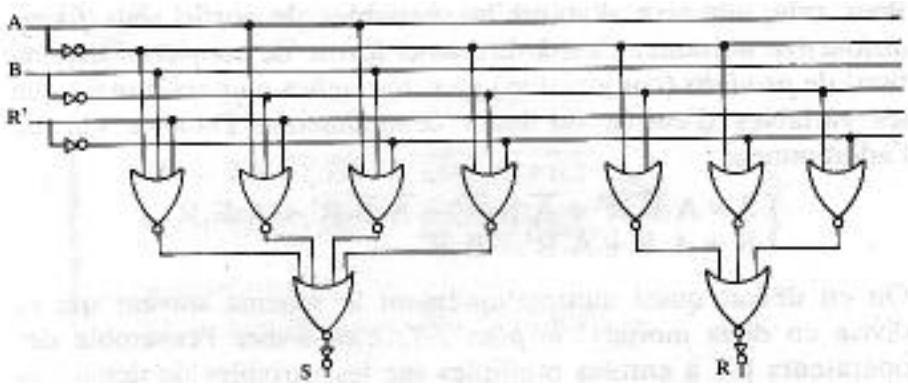
Demi additionneur :



Mise en cascade :

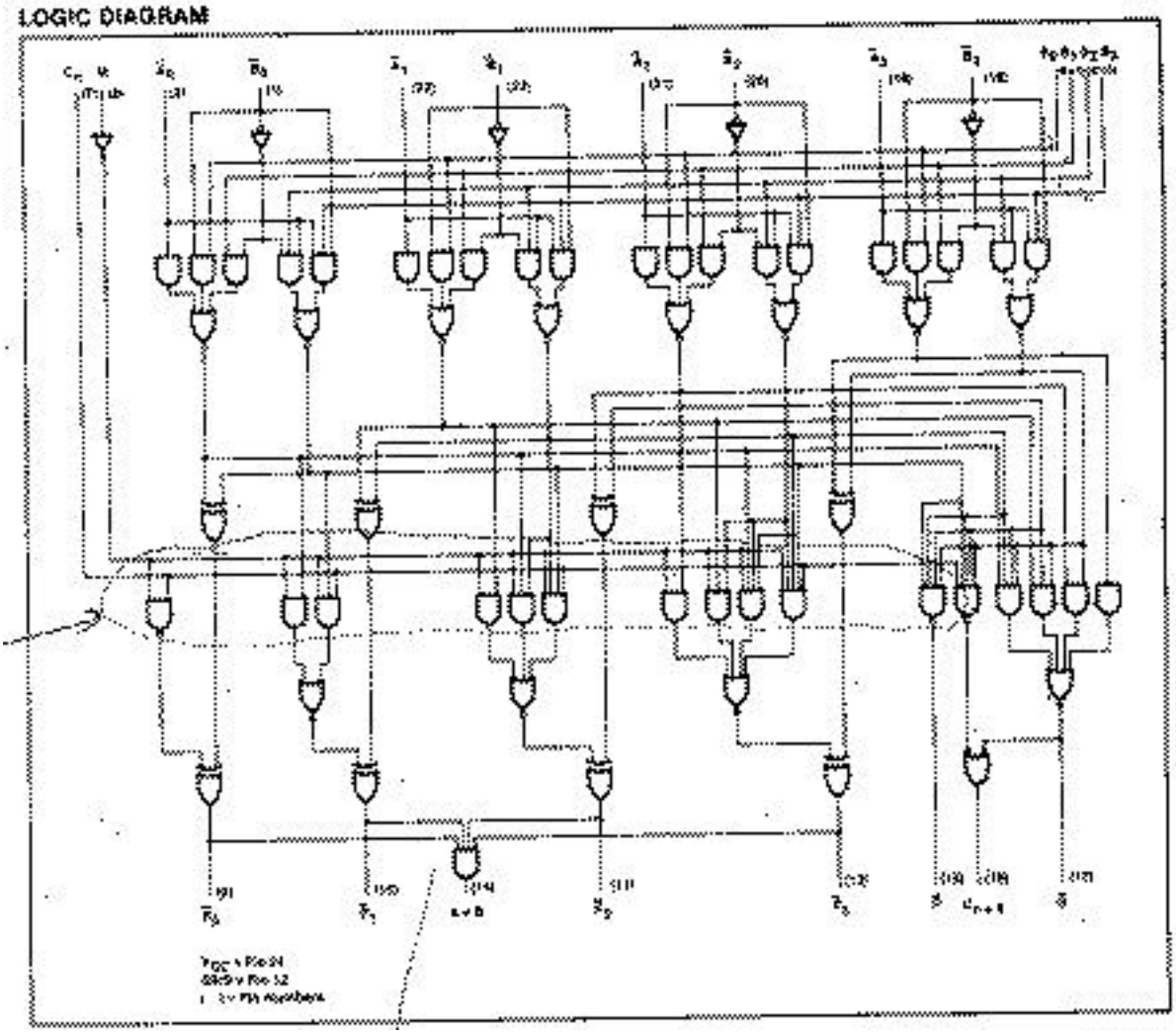


Avec seulement des ou-exclusifs :



Une UAL

SN 74181 (Texas Instruments) : 24 ns



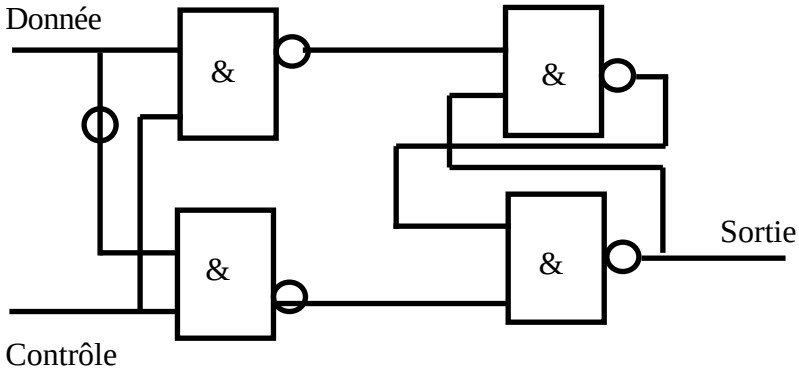
MODE SELECT INPUTS				ACTIVE HIGH INPUTS & OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M = H)	ARITHMETIC* (M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A+B}$	A + B
L	L	H	L	$\overline{A}B$	A + $\overline{B}$
L	L	H	H	Logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus AB
L	H	L	H	$\overline{B}$	(A + B) plus AB
L	H	H	L	A ⊕ B	A minus B minus 1
L	H	H	H	AB	AB minus 1
H	L	L	L	$\overline{A}$ + B	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	(A + $\overline{B}$) plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	Logical 1	A plus A*
H	H	L	H	A + $\overline{B}$	(A + B) plus A
H	H	H	L	A + B	(A + B) plus A
H	H	H	H	A	A minus 1

MODE SELECT INPUTS				ACTIVE LOW INPUTS & OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M = H)	ARITHMETIC* (M = L) (C_n = L)
L	L	L	L	$\overline{A}$	A minus 1
L	L	L	H	$\overline{AB}$	AB minus 1
L	L	H	L	$\overline{A}$ + B	A$\overline{B}$ minus 1
L	L	H	H	Logical 1	minus 1
L	H	L	L	$\overline{A+B}$	A plus (A + $\overline{B}$)
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)
L	H	H	L	$\overline{A \oplus B}$	A minus B minus 1
L	H	H	H	A + $\overline{B}$	A + B
H	L	L	L	$\overline{A}$B	A plus (A + B)
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	A$\overline{B}$ plus (A + B)
H	L	H	H	A + B	A + B
H	H	L	L	Logical 0	A plus A*
H	H	L	H	AB	AB plus A
H	H	H	L	AB	AB plus A
H	H	H	H	A	A

Le bistable (bascule, trigger, flip-flop…)

Eccles et Jordan, Radio Review, oct. 1919

Utilisation comme mémoire 1 bit :



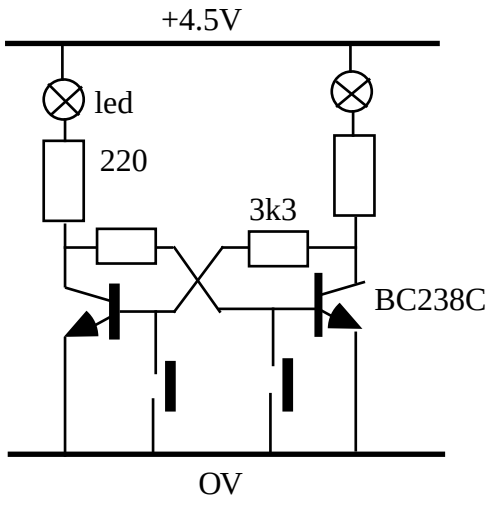
Contrôle	Donnée	Sortie
0	0	D(t-1)
0	1	D(t-1)
1	0	0
1	1	1

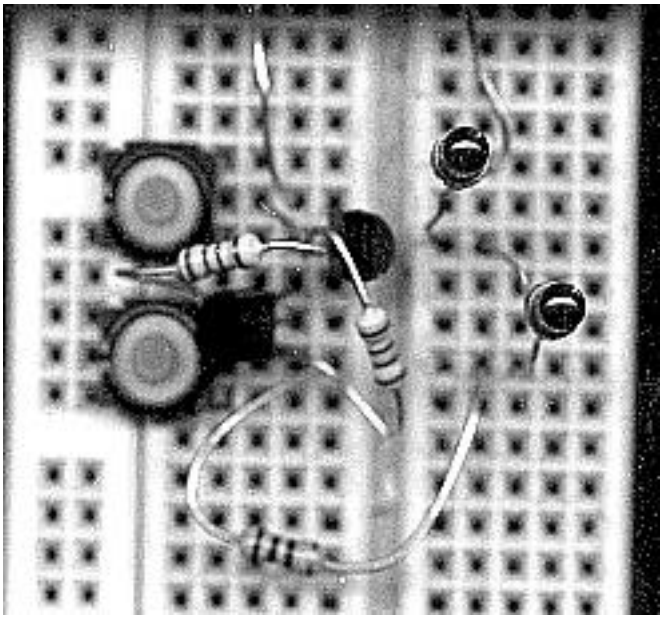
Cas particulier d'automate fini

E(t) = Entrée à l'instant t S(t) = Sortie à l'instant t
Q(t) = Etat de l'automate

S(t+1) = f(E(t), Q(t)) et Q(t+1) = g(E(t), Q(t)) (Mealy)

Exemple de réalisation





II.3 Le programme enregistré

Knuth "Von Neuman's First Computer Program" ACM Comp. Surveys, 2(4), dec. 1970.

La question du tri de données :

Il existe à l'époque des machines électro-mécaniques spécialisées très efficaces (tabulatrices)

Une méthode de résolution (parmi d'autres) :

- Rechercher le plus petit élément parmi les 10 cases

000	001	002	003	004	005	**006**	007	008	009
12	5	7	0	34	9	**1**	7	9	10

- Faire l'échange avec la première case

000	001	002	003	004	005	**006**	007	008	009
1	5	7	0	34	9	**12**	7	9	10

- Recommencer à partir de la deuxième case, etc…

	001	002	003	004	005	006	007	008	009
	5	7	0	34	9	12	7	9	10

(Méthode du tri-bulle)

MAIS : comment accéder à la position du plus petit élement puisque elle ne peut être connue à l'avance ? C'est le résultat d'un calcul…

Algorithme

```
pour i de 0 à 9 faire
        m <- i
        pour j de i à 9 faire
                si Tj < Ti alors m <-j
        fin pour
        Ti <- Tm
fin pour
```

Une solution "à la" von Neuman :

- les données

000	001	002	003	004	005	006	007	008	009
12	5	7	0	34	9	1	7	9	10

- les variables de travail

095	096	097	098	099
j<10 i<10	m	Tj<Ti	9	0

- le programme

100	ING	**0**	**1**	97
104	BRA	+0…	8	97
108	CPY	102	///	96
112	INC	+0…	1	102
116	ING	102	98	95
120	BRA	-0…	20	95
124	**CPY**	**95**	///	**133**
128	**CPY**	**101**	///	**135**
132	**CPY**	**xxx**	///	**xxx**
136	INC	+0…	1	101
140	ING	101	98	95
144	BRA	-0…	44	95
148	HLT	+0…	0	///

Impact sur l'organisation

Le programme n'est plus accessible en séquence : il faut un compteur pour mémoriser la position courante (et l'incrémenter)

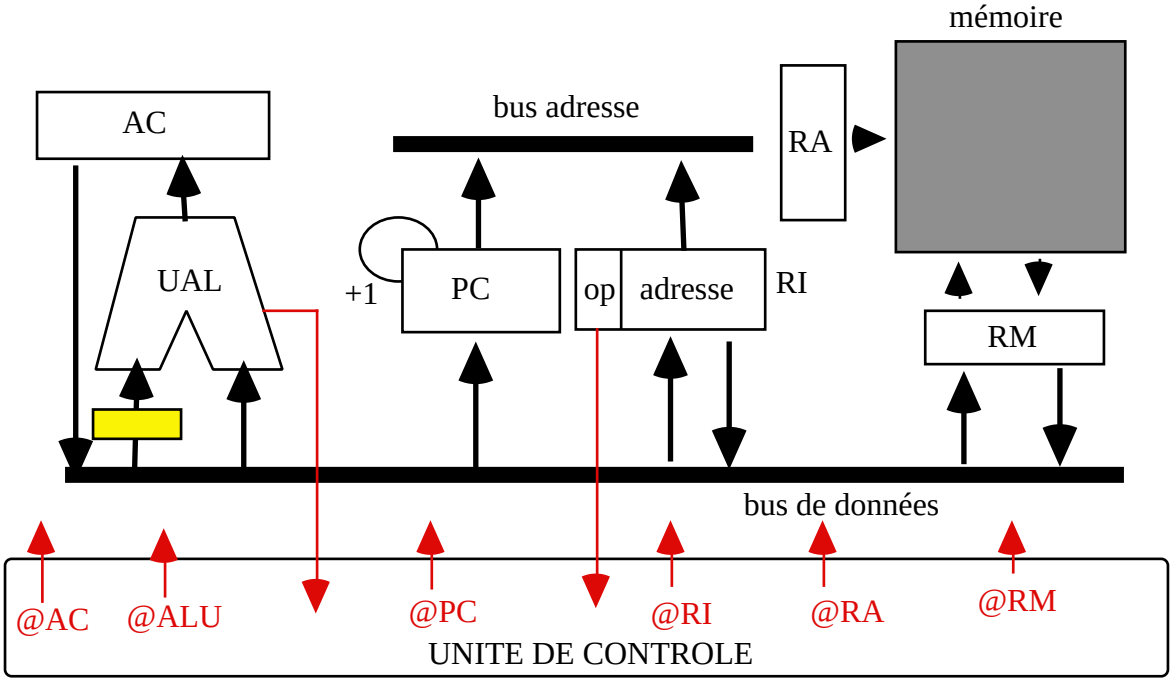
Compteur ordinal, compteur de programme, PC …

Il faut revoir le codage des instructions pour qu'elles coincident avec celui des données :

Il faut mémoriser quelque part l'instruction courante : registre d'instruction (RI) si nbre bits adresses > nbre bits données

Favoriser des instructions "courtes", à 0 ou 1 paramètre
=> Opérations de calculs relatives à AC (accumulateur)
=> Créer d'autres zones mémoires specialisées proches de l'UAL (registres)

RQUE : Il faut aussi mémoriser localement le résultat produit par l'unité arithmétique (sinon, oscillations)



Codage des instructions

Données sur 40 bits => instruction aussi ?

adresse opérateur = 38 bits si 16 instructions au plus
2^38 = 2700 milliards de cases possibles (au lieu de 4000)

Si nbre bits données >> nbre bits adresse : plusieurs instructions par mot mémoire

Si nbre bits données < nbre bits adresse : plusieurs mots mémoire par instructions

Exemple naïf de compromis :

- mot mémoire de 64 bits => entiers de 0 à 1.8*10^19

- 2 instructions par mot : 32 bits pour une instruction

- adresse sur 27 bits (134 M de cases possibles, soit 1 Go)

- 5 bits code opération => 32 instructions

En pratique :

- Plusieurs types de données : des nombres réels (64 bits) ≠ entiers (32 bits) ≠ caractères (8 bits) et des opérations arithmétique/logiques différentes

- Plusieurs types d'instructions, avec 0, 1, 2 paramètres

=> très nombreuses variantes, selon l'époque (contraintes technologique) et le constructeur (type de machine voulue)

Modes d'adressage

La méthode de von Neuman est dure à employer pour de gros programmes.

Elle se prête difficilement à une génération automatique des instructions machines

Elle est maintenant totalement déconseillée !!

Adressage indirect :

Le *contenu* d'une case mémoire indique *l'adresse* de la donnée recherchée

Il faudra donc deux accès mémoire pour obtenir la donnée

On peut également appliquer le concept aux branchements

Variantes (+ rapides d'exécution)

• Adressage indexé : un registre spécial (INDX) en plus de l'ACC stocke une adresse qui est ajoutée à celle de l'instruction. INDX peut être modifié par une instruction spéciale.

=> utile pour itérer dans un tableau de données

• Adressage basé : un registre stocke la partie haute du mot adresse et l'instruction, la partie basse

=> utile pour découper la mémoire en "pages"

• On peut les combiner à l'indirection : pour itérer dans un tableau d'adresses, par ex.

Procédures & piles

Que faire quand un traitement est à appliquer plusieurs fois dans un programme ?

Avec des cartes perforées, il suffit de les dupliquer autant de fois que nécessaire

… Mais cela suppose qu'on peut savoir à l'avance sur quelles données (et donc leur adresse) le traitement dupliqué s'applique

=> utilisation du branchement indirect (instruction GTI)

000 -- debut du programme
…
100 -- calcul de sinus(x) avec x stocké dans case 5000
104 CPY 5000 499 -- passage de la valeur
108 SET 116 498 -- passage de l'adresse retour
112 GTO 500 -- Saut sans condition à l'instruction 500
116 -- utilisation de sin(x)
…
200 -- calcul de sinus(y) avec y stocké dans case 5001
204 CPY 5001 499 -- nvelle valeur
208 SET 216 498 -- nvelle adresse retour
212 GTO 500 -- Saut sans condition à l'instruction 500
216 -- utilisation de sin(y)
…
300 HLT -- fin du programme
…
…
498 <<stockage adresse retour>>
499 <<stockage paramètre>>
500 -- debut procedure sinus
504 --utilisation de la case 499 pour calculer le sinus
…
568 GTI 498 -- retour à l'envoyeur

On peut "automatiser" la technique précédente grâce au compteur de programme :

Une nouvelle instruction pour l'appel

SBR <adresse> :
- stockage dans un registre spécial SP du CP courant
- GTO <adresse>

+ Une nouvelle instruction pour le retour

RTN :
- recopie du SP dans le PC

Problèmes :

- On n'a pas réglé la question de la transmission du paramètre, mais la technique précédente reste valable

- Plus génant : que faire la procédure en appelle une autre ?? l'adresse de départ est perdue avec cette méthode

Solution :

Le SP référence en fait une zone de la mémoire où déposer le CP courant ("stack pointer")

SBR <adresse> :
    incrémenter SP
    CPY CP (SP)
    GTO adresse

RTN :
    CPY (SP) PC
    décrémenter SP

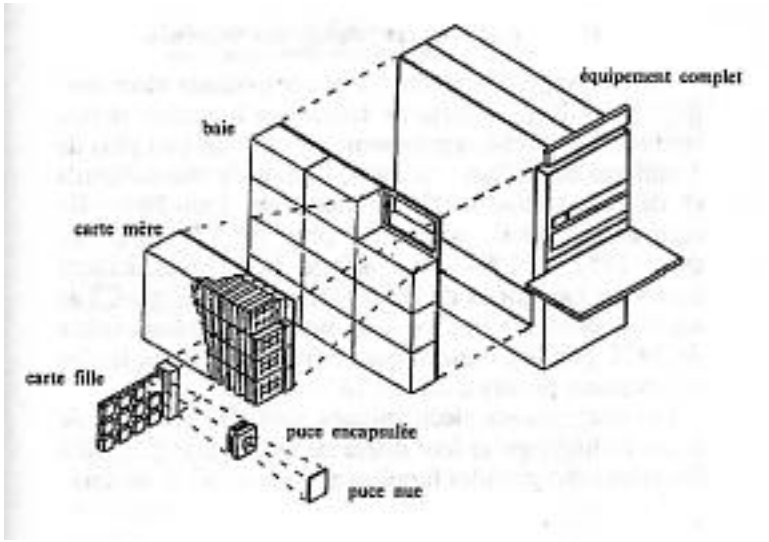
Acte III : 1973

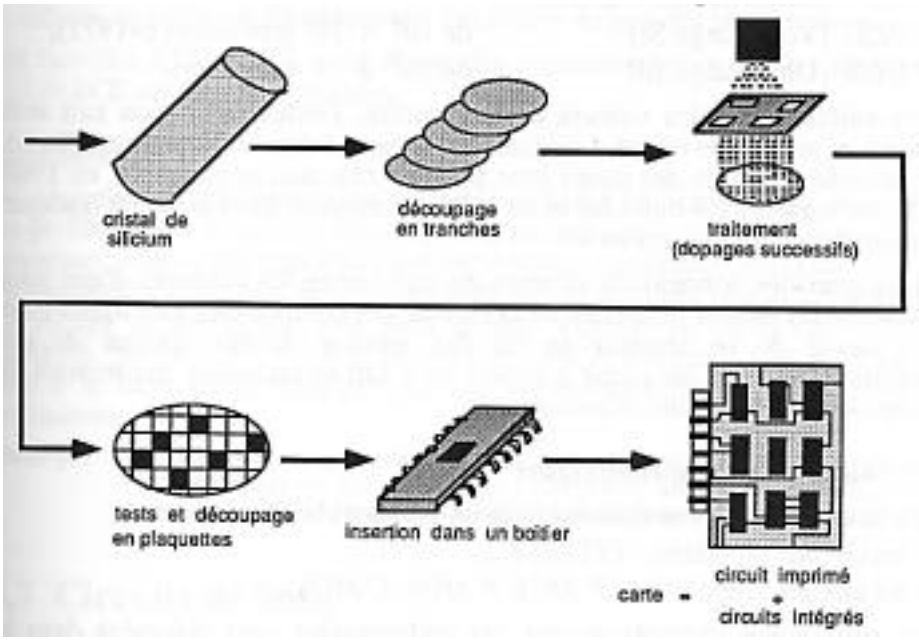
L'ordinateur de bureau Alto

Laboratoire d'informatique du XEROX PARC
B. LAMPSON, Ch. THACKER, R. TAYLOR

- OS temps partagé + interface graphique multi-fenêtres
- Ecran N&B 606 x 808 pixels (80 ppi)
- Clavier séparé, reconfigurable, mesure de force et durée
- Souris 3 boutons
- 2 disques durs de 3 Mo pour le stockage local
- Ethernet
- Imprimante laser

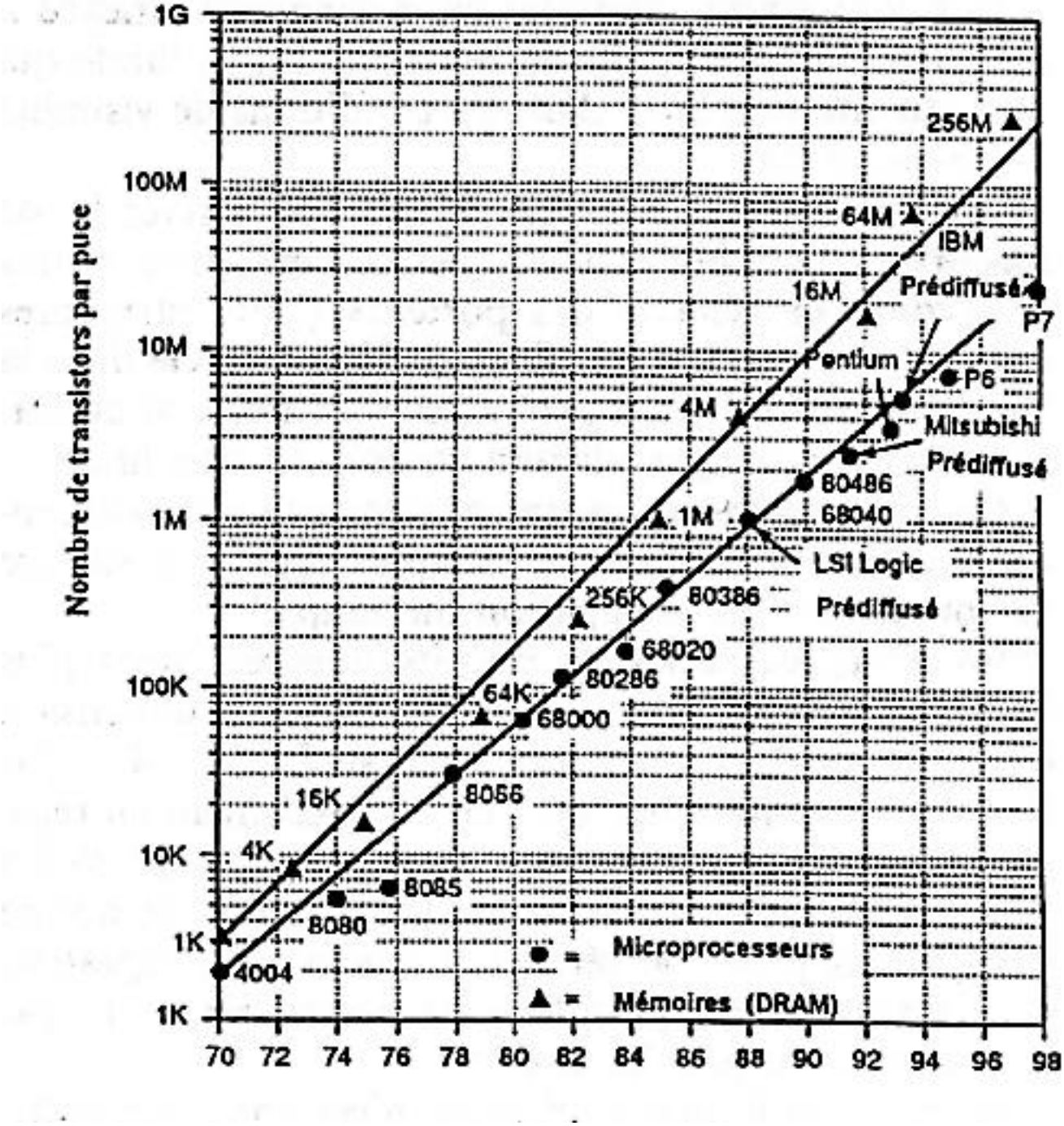
2. Progrès de l'intégration





Technologie	Vitesse	Consommation	Densité
TTL	grande	grande	petite
ECL	très grande	très grande	petite
N-MOS	moyenne	grande	très grande
P-MOS	petite	grande	grande
C-MOS	grande	très petite	grande

La « loi » de Gordon Moore

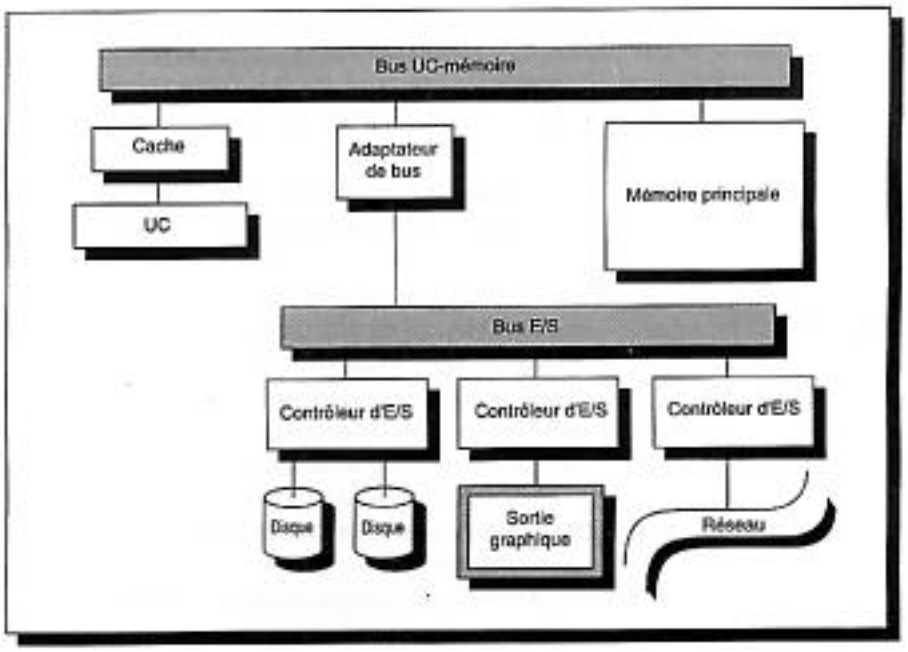


Densité mémoire : x 1.5 / an
Densité microprocesseur : x 1.35 / an

Date	Finesse gravure	Nbre transistors	Surface puce	Nbre E/S
1998	0.25 µm	256 M	320 mm2	1500
2001	0.18	1 G	500	2000
2004	0.12	4 G	1000	3500

3. La communication

Le couple processeur/mémoire est inutilisable sans les dispositifs d'entrée-sortie



Ce sont les "périphériques" qui servent à départager les différentes gammes d'ordinateurs :

- mainframe / mini : un gros ordinateur peut supporter un très grand nombre de terminaux et de disques

- mini / workstation : une station a toujours un écran graphique, un clavier, une souris et ± 1 utilisateur

- serveur fichier / workstation : le serveur a plus de disque et dispositif de sauvegarde, mais pas d'écran, clavier, souris

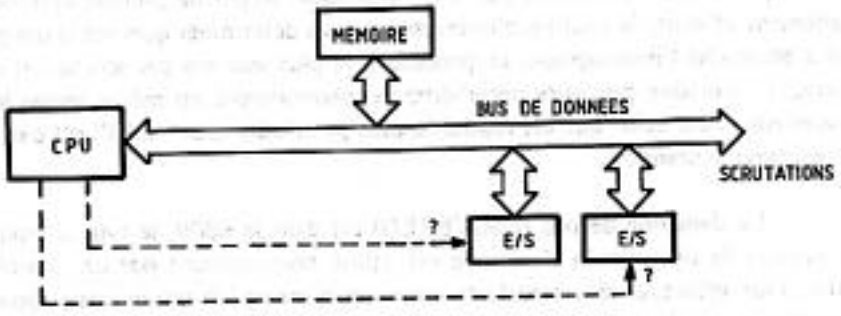
- station / PC : la station est toujours connectée en réseau

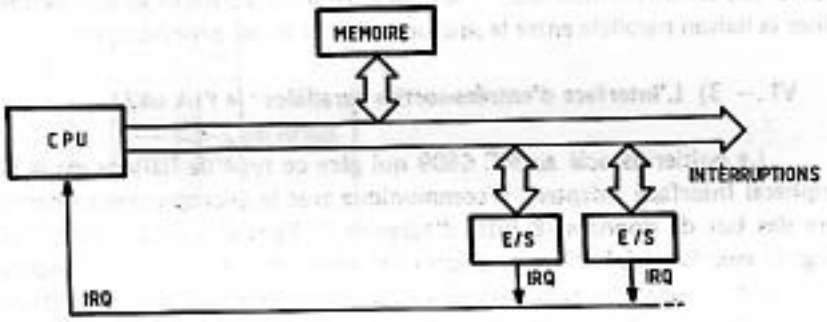
Problème : grande disparité dans les débits

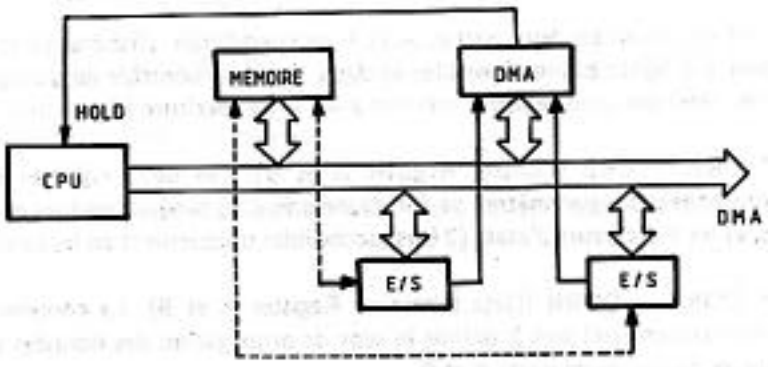
Interaction avec les périphériques

Les périphériques doivent lire/écrire dans la mémoire centrale : conflit avec le processeur

3 modes d'interactions - qui peuvent co-exister :







Les bus

L'organisation en bus permet d'augmenter le nombre de périphériques (≠ étoile)

Fonctionnement synchrone ou non

- synchrone possible sur petites longueurs (ou débit faible)
- sinon: échange selon un *protocole*

Arbitrage

- prise en compte des ≠ priorités entre les dispositifs (ex : daisy chain)
- maintient de l'équité
- un goulot d'étranglement potentiel

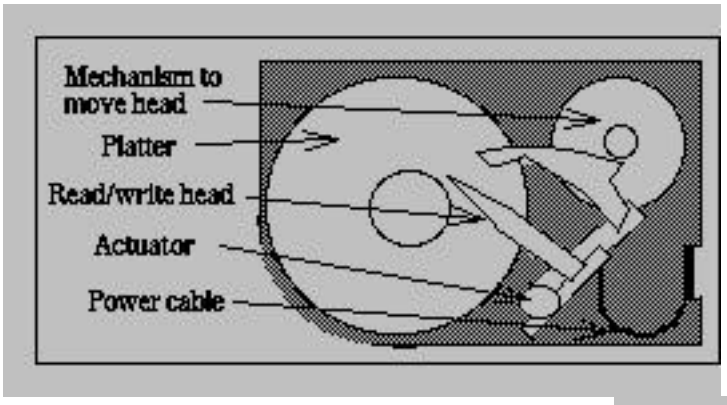
Profusion de produits et de normes :

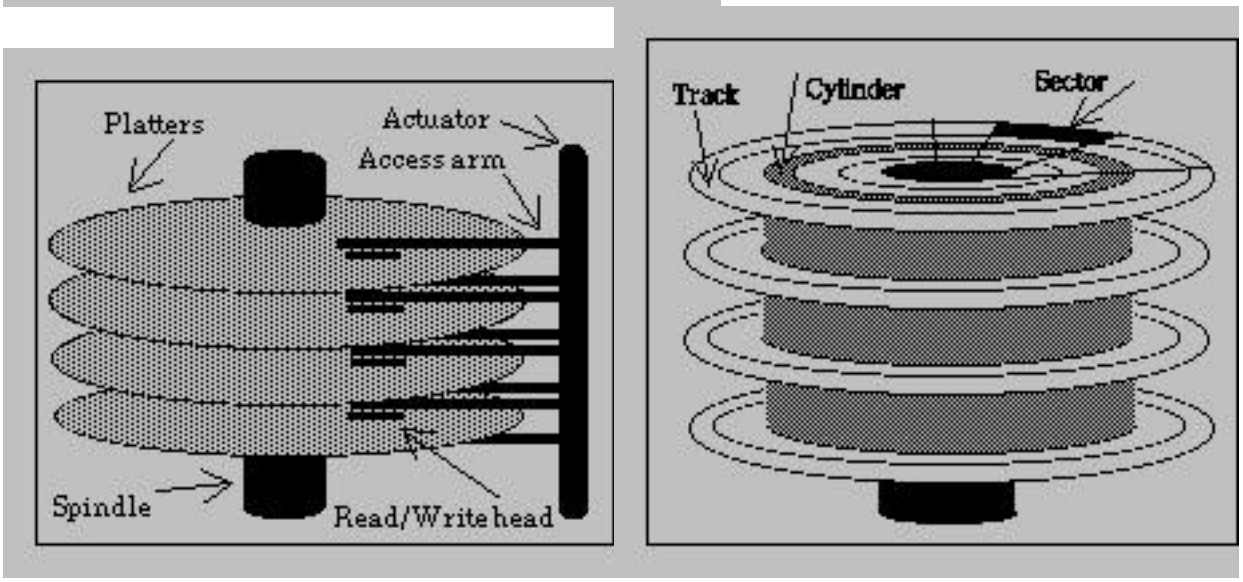
	bus VME	NuBus	SCSI
type de bus	fond de panier	fond de panier	E/S
largeur (signaux)	128	96	8
multiplexage adresses/données ?	non	oui	hors sujet
largeur données	16-32	32	8
Arbitrage	daisy chain	auto-sélection	autosélection
Horloge	Asynchrone	Synchrone	L'un ou l'autre
Débit vers mémoire 150 ns (1 mot)	12.9 Mo/s	13.3 Mo/s	5 Mo/s max
idem en continu	13.6 Mo/s	26.4 Mo/s	Inchangé
Nbre max dispositifs	21	16	7
Long max bus	0.5 m	0.5 m	25 m

Ex. des PC et compatibles : bus ISA, EISA, VESA, PCI, AGP, PCMCIA…

Les disques

Je pense que la "silicon valley" est mal nommée […]
Ils auraient dû [la] renommer "Iron Oxyde valley"
A. Hoagland, cité par Patterson & Hennessy





Prix du Mo sur disques durs

1988 : 11.54 (USD)
1990 : 06.86
1992 : 03.00
1994 : 00.71
1996 : 00.18
1998 : 00.07 (prevision)
2000 : 00.03 (")

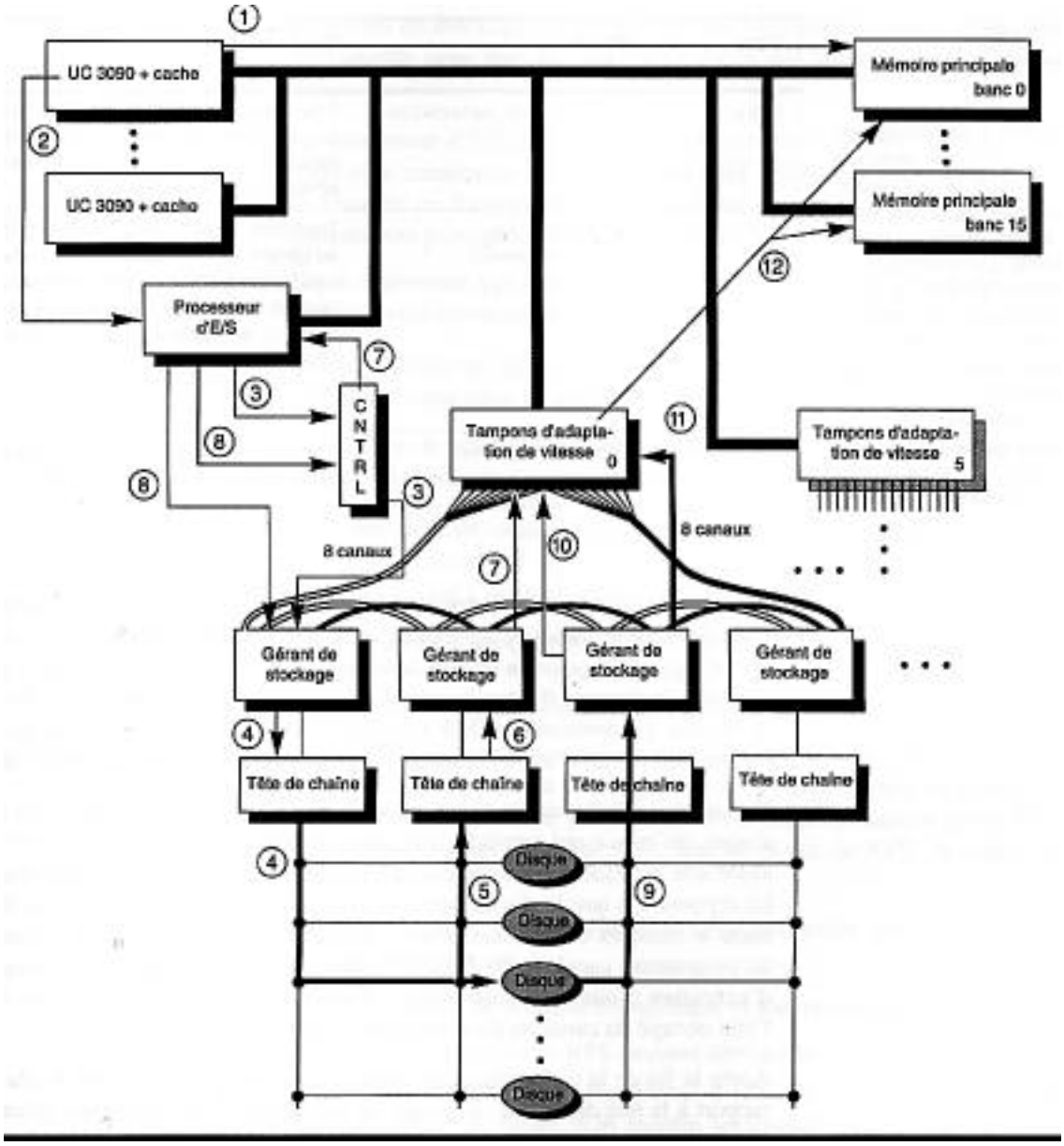
Source : Courrier international n° 378 29/1/97 p. 36
Dans le même journal : IBM (Centre d'Almaden) : 11 Go sur 1 pouce-carré

Encore indispensables …

Exemple (extrême…)

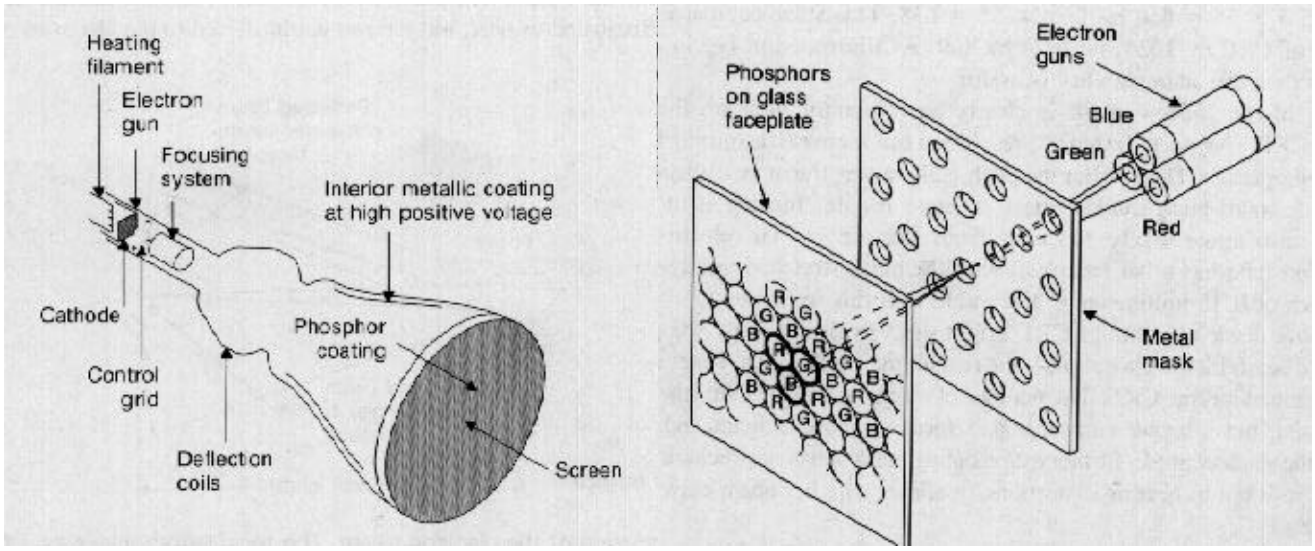
Gros système IBM (circa 1990 ?)

Unité centrale 3090 (1 à 6 processeurs 30 MIPS, cache 64Ko)
Sous système de stockage 3990 (1 à 6 groupes de 64 disques 3390 de 6 à 540 Go)



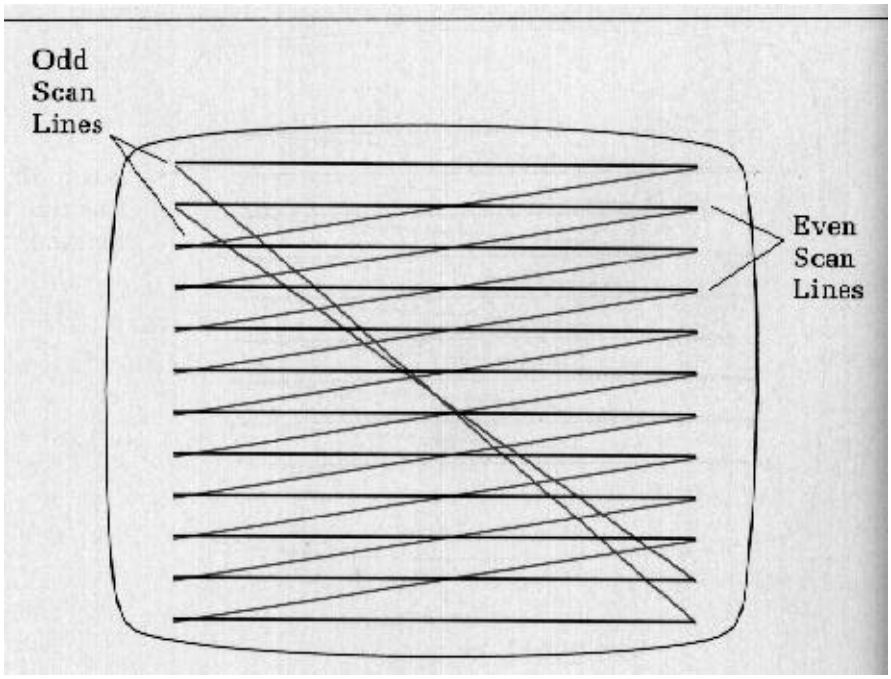
L'affichage graphique

• Un tube vidéo couleur :



[FOLEY] p. 155 et p. 159

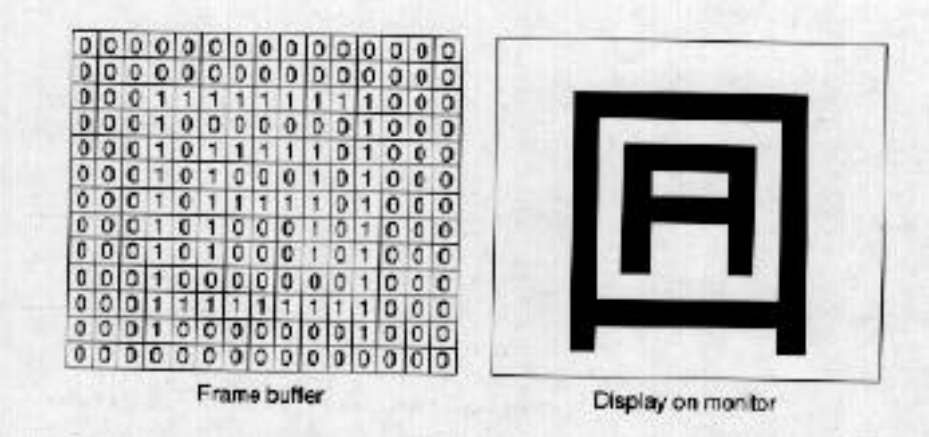
• Balayage : entrelacé ou non



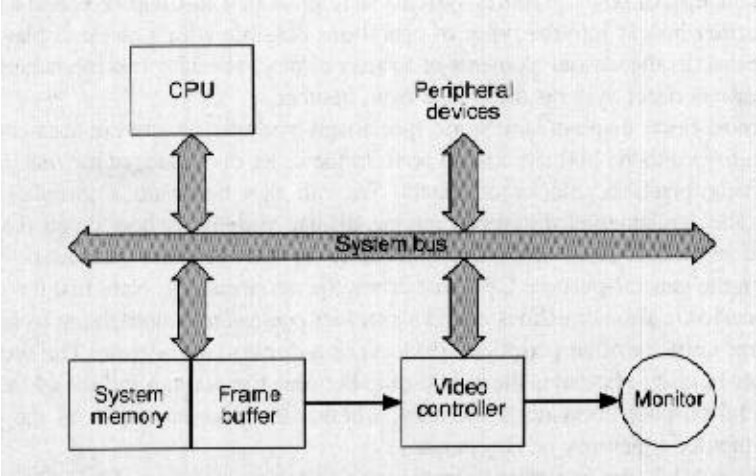
[HILL] p. 30

• Paramètres : résolution (points/pouce), taux de rafraichissement (ex : 75Hz)

- Le contrôleur vidéo



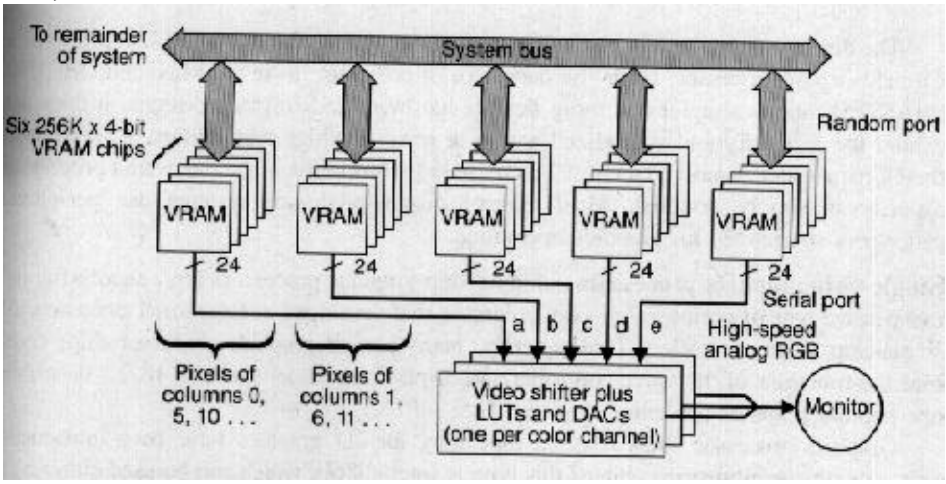
[SUN] p. 31



[FOLEY] p. 166

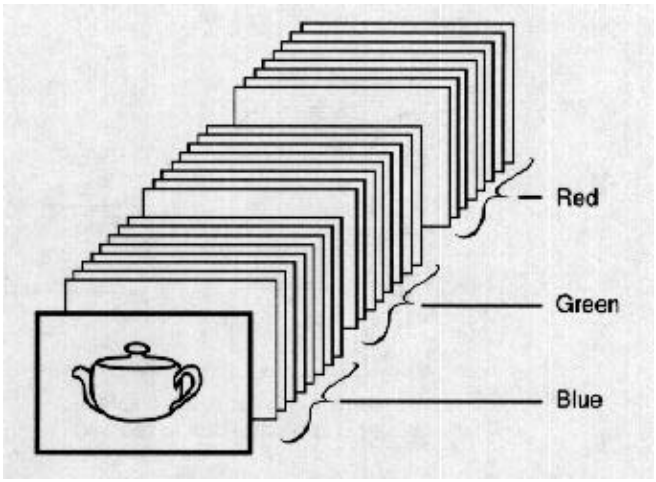
- La mémoire vidéo

  VRAM, Texas Instruments 1983

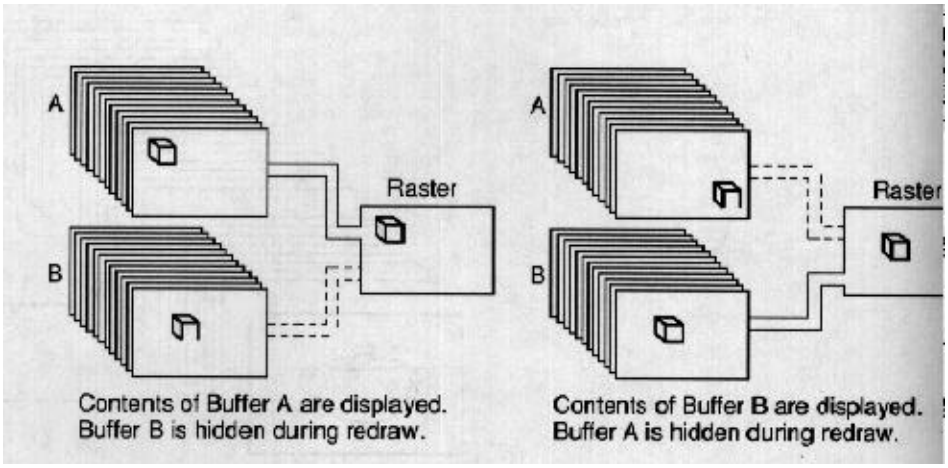


- Organisation de la mémoire vidéo

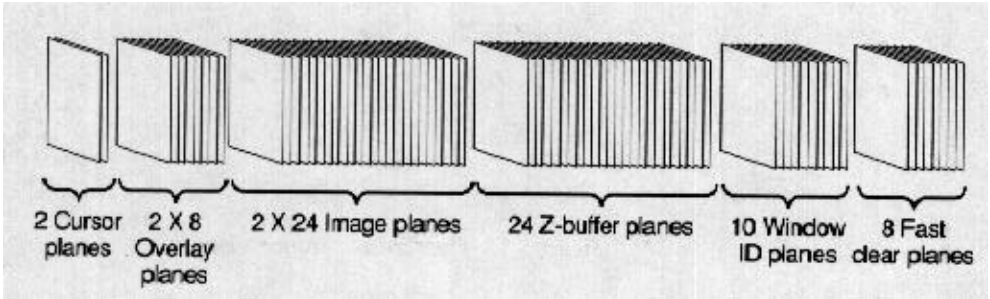
Exemple : 8 bits par couleur primaire = 24 bits



Une astuce pour l'animation : le "double buffering"



Un système complet (sur 108 bits !!) :

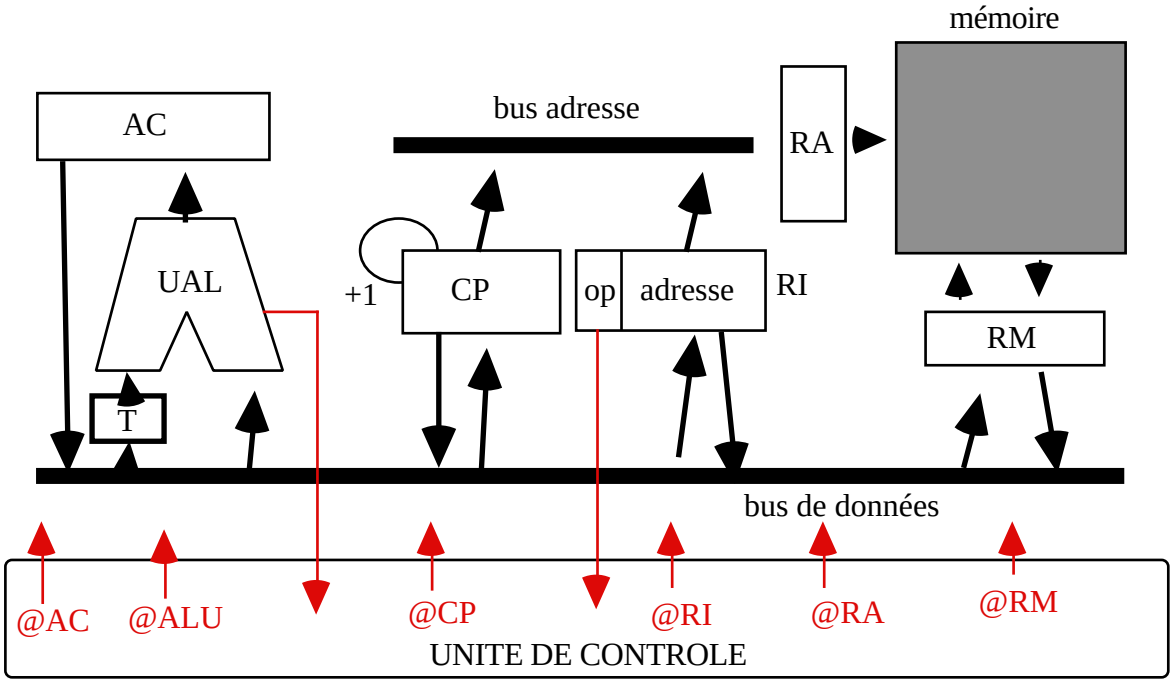


Double buffers + Z-buffer intégré
+ gestion du curseur, des menus déroulants, des fenêtres, de l'effacement des plans et du fond d'écran

VARI - Exercice Dirigé
Architecture des ordinateurs

Exercice 1 - Un processeur minimal

Reprenons l'architecture d'un processeur minimal vue en cours.



Rappeler le rôle des différents organes dans le schéma ci-dessus

Que manque t'il surtout dans cette architecture minimale ?

On suppose que tous les registres sont de 64 bits, ainsi que les mots mémoires.

Quelle est la taille de l'espace mémoire adressable ?

Quels sont les nombres manipulables par l'UAL ?

Comment peuvent être codées les instructions de ce processeur ? Quel est l'inconvénient de cette approche et comment le résoudre ?

Exercice 2 - Exemple de programme

La machine ci-dessus dispose d'instructions de trois types, qui sont toutes codées sur un mot-mémoire de 64 bits.

- des instructions sans opérande :

```
RAZ : (AC) := 0
INC : (AC) := (AC) + 1
DEC : (AC) := (AC) - 1
HALT : arrêt du programme
```

- des instructions à 1 opérande :

```
SET valeur : (AC) := valeur
LOAD case : (AC) := (case)
STORE case : (case) := (AC)
```

```
ADD case : (AC) := (AC) + (case)
SUB case : (AC) := (AC) - (case)
EQU case : si (AC) = (case) alors (AC) := 0
INF case : si (AC) < (case) alors (AC) := 0
BRA valeur : si (AC) = 0 alors (CP) := (CP) + valeur
```

- des instructions d'adressage indirect :

```
LOADi case : (AC) := ((case))
STOREi case : ((case)) := (AC)
ADDi case : (AC) := (AC) + ((case))
SUBi case : (AC) := (AC) - ((case))
EQUi case : si (AC) = ((case)) alors (AC) := 0
INFi case : si (AC) < ((case)) alors (AC) := 0
BRAi case : si (AC) = 0 alors (CP) := (CP) + (case)
```

On veut écrire un programme dont la seule fonction est de se dupliquer dans la mémoire, par saut de 1000 cases. On supposera que la position en mémoire de la première instruction est connue. Le programme s'arrête quand toute la mémoire est parcourue.

Ecrire ce programme en supposant que ses données ne sont pas dupliquées.

Que manque-t'il au jeu d'instruction pour que les données soient elles aussi dupliquées ?

Nommer un type de programme utilisant ce mécanisme. Quels sont les problèmes posés par ce type de programme ?

Exercice 3 - Chronogramme d'exécution

Décrire le séquencement de l'instruction ADD 155

On indiquera le déroulement des phases :

- de recherche de l'instruction courante ("fetch"),

- de décodage ("decode"),

- d'exécution ("execute").

Chaque étape correspond à une action élémentaire (micro-instruction) de l'unité de contrôle.

Décrire le séquencement de l'instruction ADDi 155

Combien de cycles d'horloge sont maintenant nécessaires ?

Quelle valeur doit-on choisir pour le temps de cycle élémentaire?