

Acte II : 1945

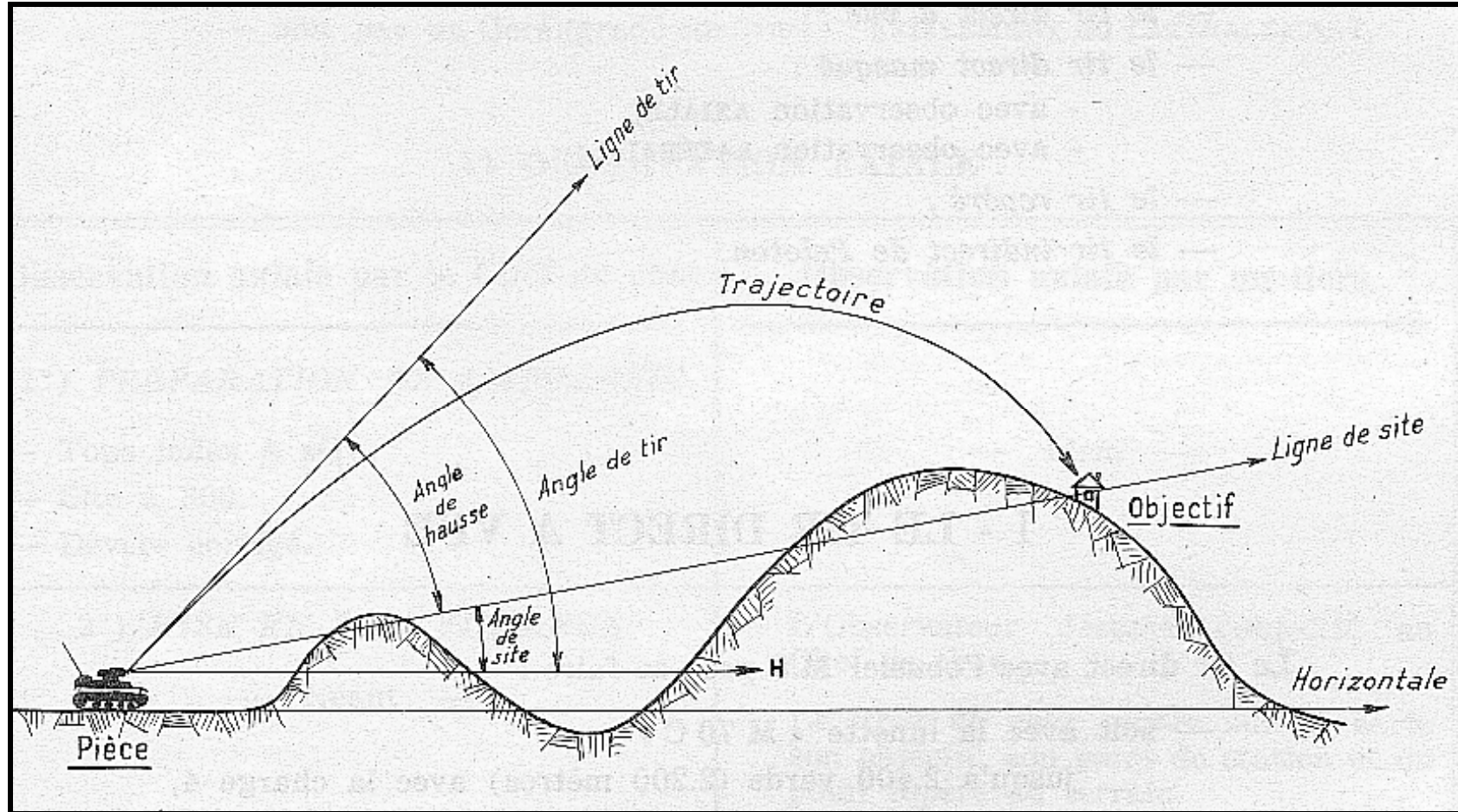


TABLE DE TIR DE L'OBUSIER DE 75 millimètres

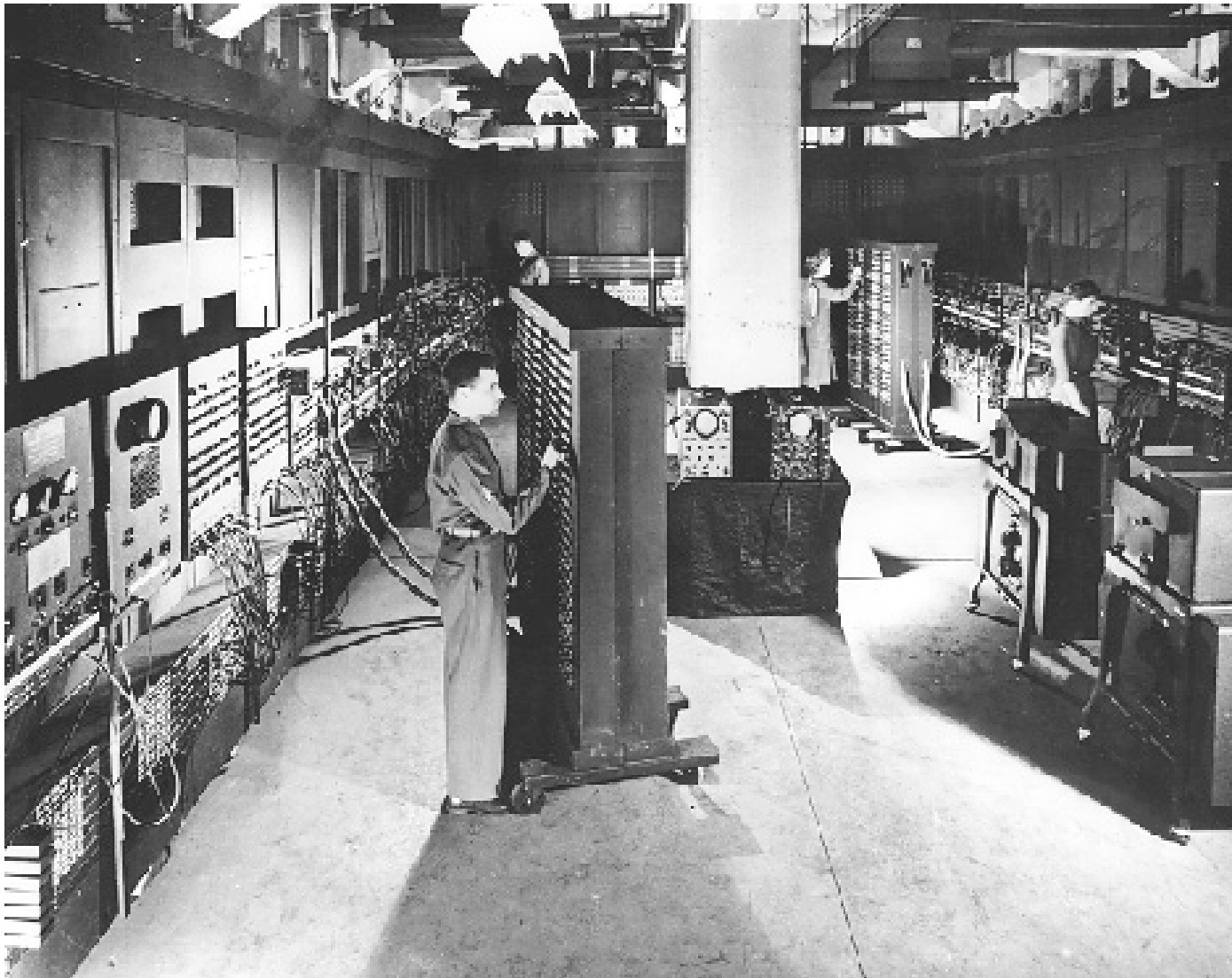
DISTANCE		CHARGE 1 (1)		CHARGE 2 (2)		CHARGE 3 (3)		CHARGE 4 (4)	
Mètres	Yards	Hausse millièmes (a)	Fourchette millièmes (b)	Hausse millièmes (c)	Fourchette millièmes (d)	Hausse millièmes (e)	Fourchette millièmes (f)	Hausse millièmes (g)	Fourchette millièmes (h)
0	0	- 5	0	- 4	0	- 3	0	0,6	0
91	100	+ 5	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 interrupteurs

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

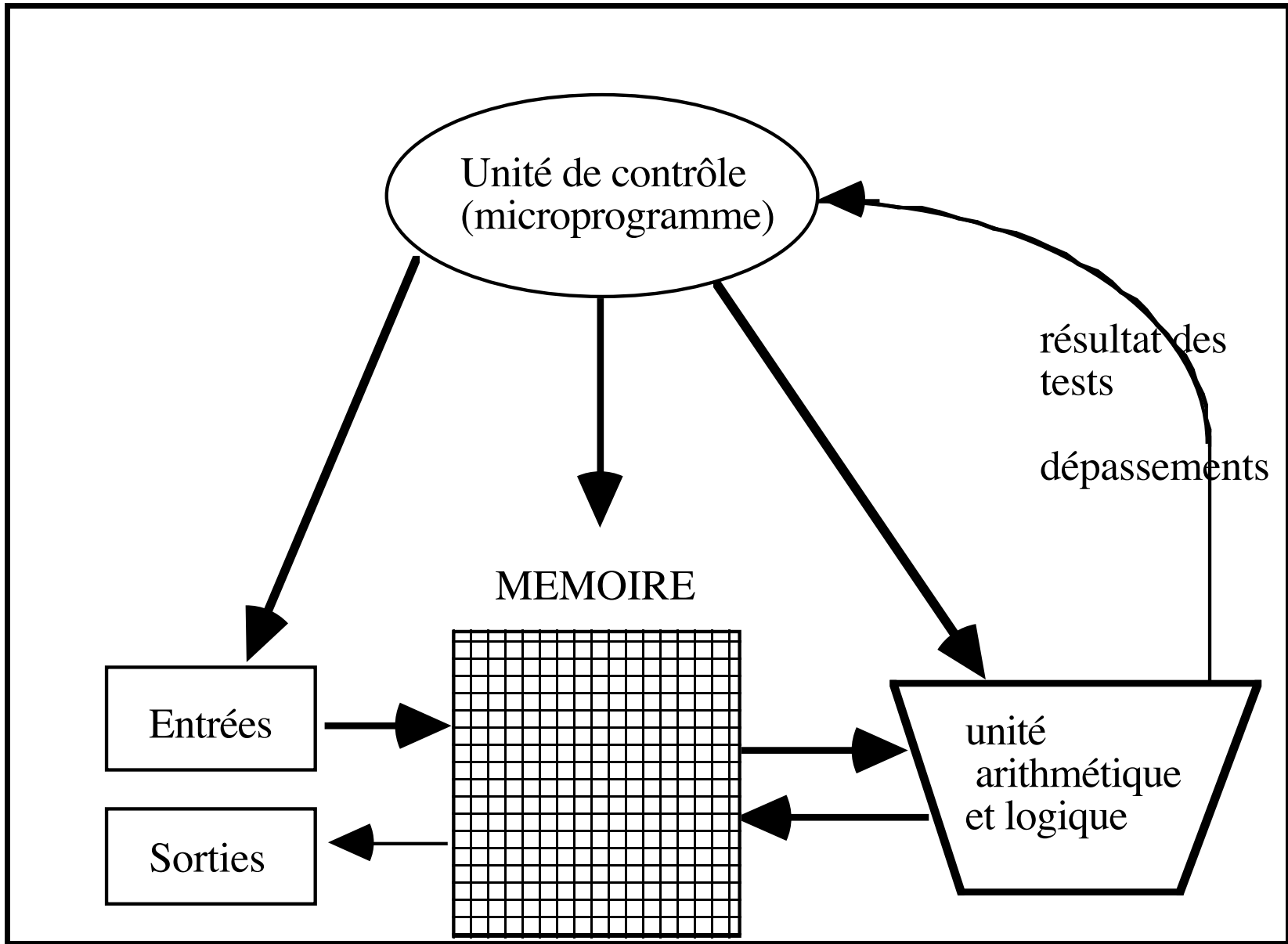
Une machine binaire (calculs et mémorisation)

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.

Règle d'Amdahl/Case : un système est équilibré, 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

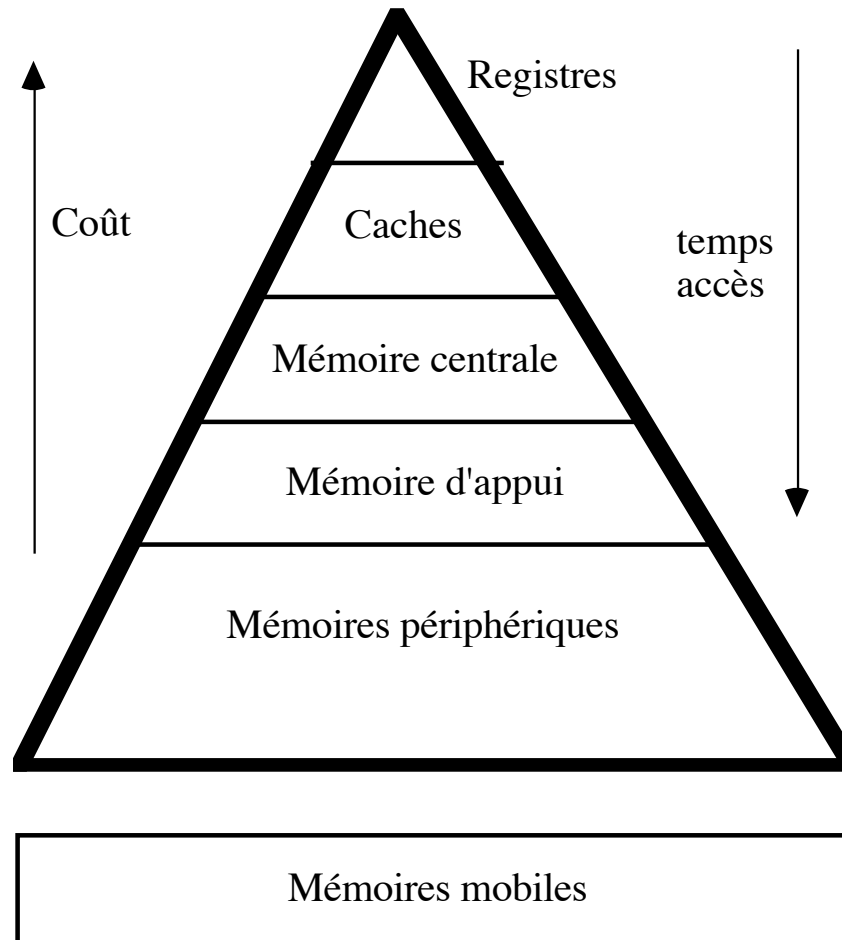
Duplication "massive" :

- 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 équivalent à 2 m² de tores

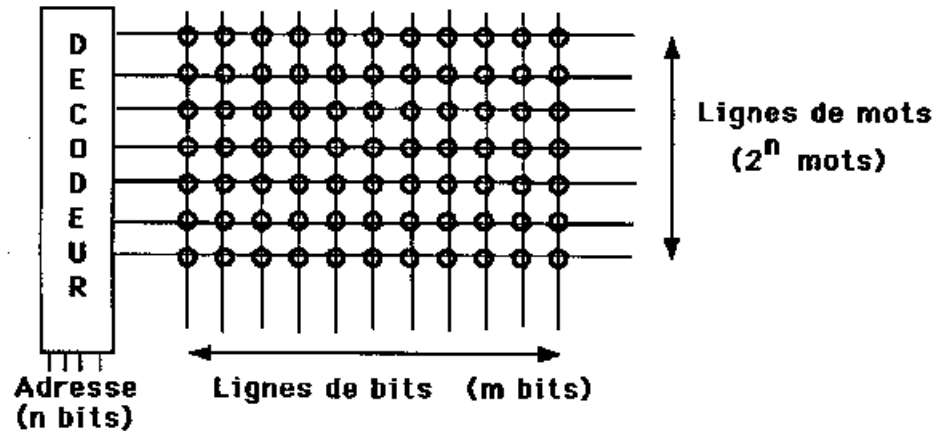
Hiérarchie de mémoires

Compromis toujours nécessaire 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

- REPROGRAMMABLE ROM : EPROM (uv)
EEPROM

- Accès séquentiel
 - cartes perforées (ROM)
 - bande magnétique (RAM)
- Accès cyclique
 - tambour électrostatique
 - 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 ISO Latin1)

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

1 Mo (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 (\neq tps de cycle) : en secondes

1 ms (milli) = 10^{-3} s

20 ms : tps accès maximum pour un relais

1 μ s (micro) = 10^{-6} s = tps accès maximum pour un tore

1 ns (nano) = 10^{-9} s

100 ns = temps d'accès DRAM

5 ns = temps d'accès bascule élémentaire

1 ps (pico) = 10^{-12} s

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 \Leftrightarrow 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 :

$$\begin{aligned} & 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \\ & = 2^6 + 2^1 + 1 = 64 + 2 + 1 \\ & = 67 \end{aligned}$$

Addition binaire $107 + 18 = 125$

$$107 = 1101011$$

$$18 = 0010010$$

$$125 = 1111101$$

Multiplication binaire $107 * 18 = 1926$

$$\bullet\bullet\bullet\bullet 87654321$$

rang

$$\bullet\bullet\bullet\bullet 1101011\bullet$$

1er multiplicande décalé à 2

$$1101011\bullet\bullet\bullet\bullet$$

2ème multiplicande décalé à 5

$$11110000110$$

addition des deux donne le total

Fonctions logiques

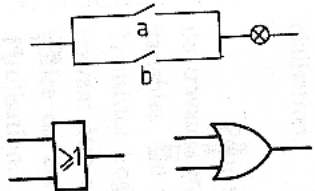
4 fonctions logiques pour une variable

16 fonctions pour 2 variables

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

$y = a + b$

nom: fonction OU, OR, réunion

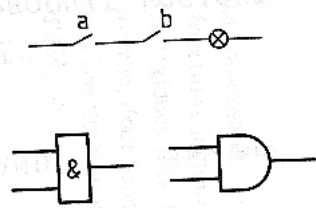


logigrammes

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

$y = a \cdot b$

nom: fonction ET, AND, intersection

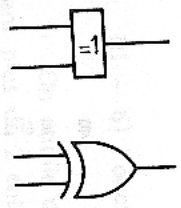


logigrammes

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$y = a \oplus b$

nom: fonction XOR

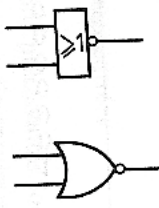


logigrammes

a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

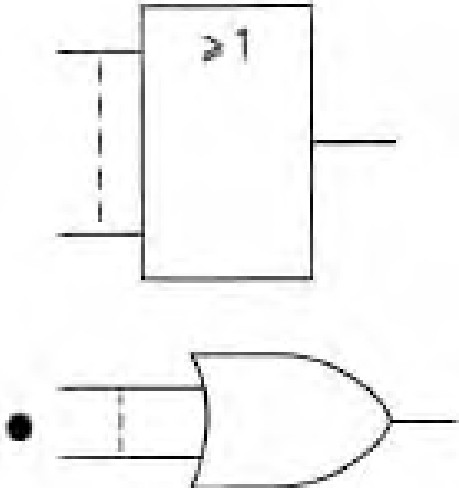
$y = \overline{a + b}$

nom: fonction NON-OU, NOR



logigrammes

Lois de Morgan : $\text{non}(A \text{ et } B) = \text{non}(A) \text{ ou } \text{non}(B)$
 $\text{non}(A \text{ ou } B) = \text{non}(A) \text{ et } \text{non}(B)$

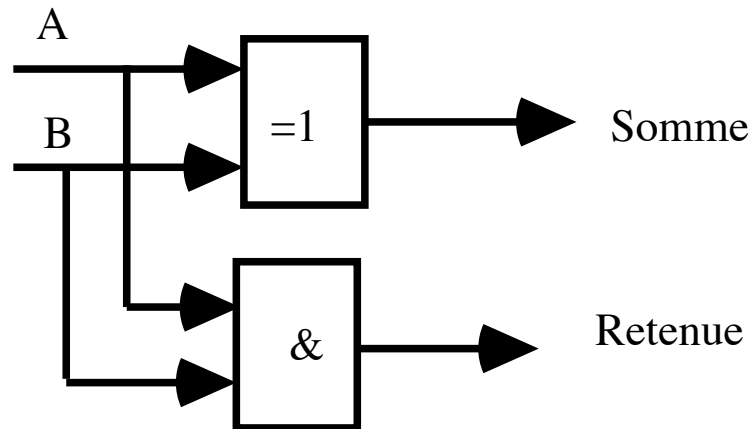
No	Symbol	Description
5.1-1		<p>OR element, general symbol</p> <p>The output stands at its 1-state if and only if one or more of the inputs stand at their 1-states.</p> <p>NOTES:</p> <ol style="list-style-type: none">1 — “≥ 1” may be replaced by “1” if no confusion is likely.2 — The distinctive-shape symbol is, according to IEC Publication 617, Part 12, not preferred, but is <i>not considered to be in contradiction</i> to that standard. Its use in combination to form complex symbols (for example, use as an embedded symbol) is discouraged.

Circuit additionneur

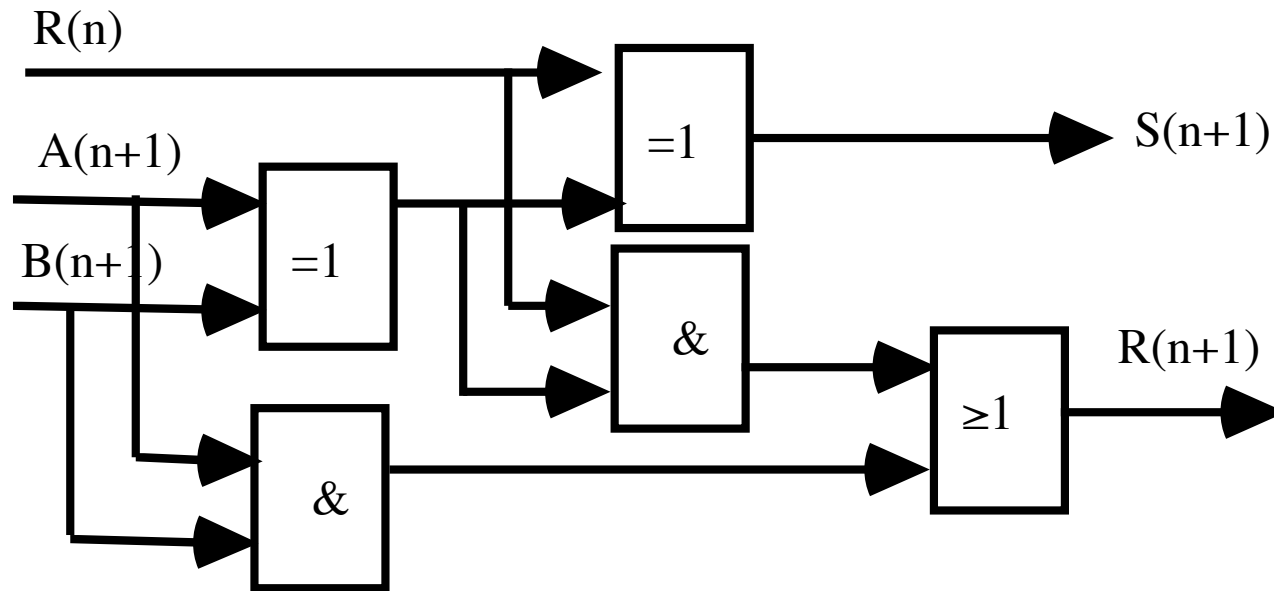
Table de vérité de la somme $S = A + B$

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

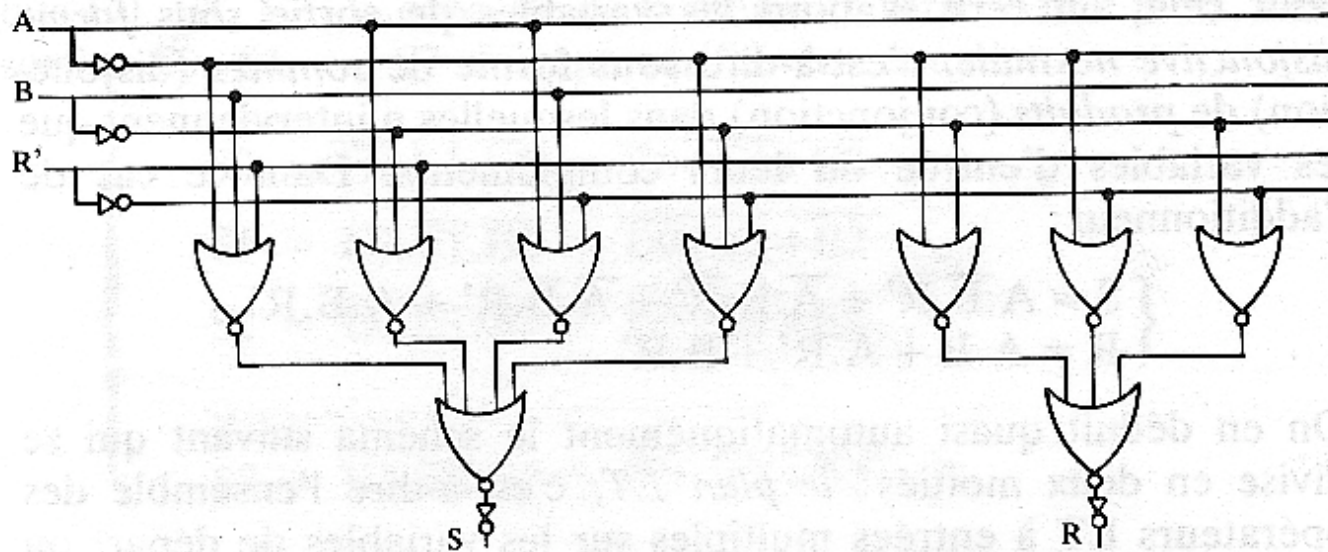
Demi additionneur :



Mise en cascade :



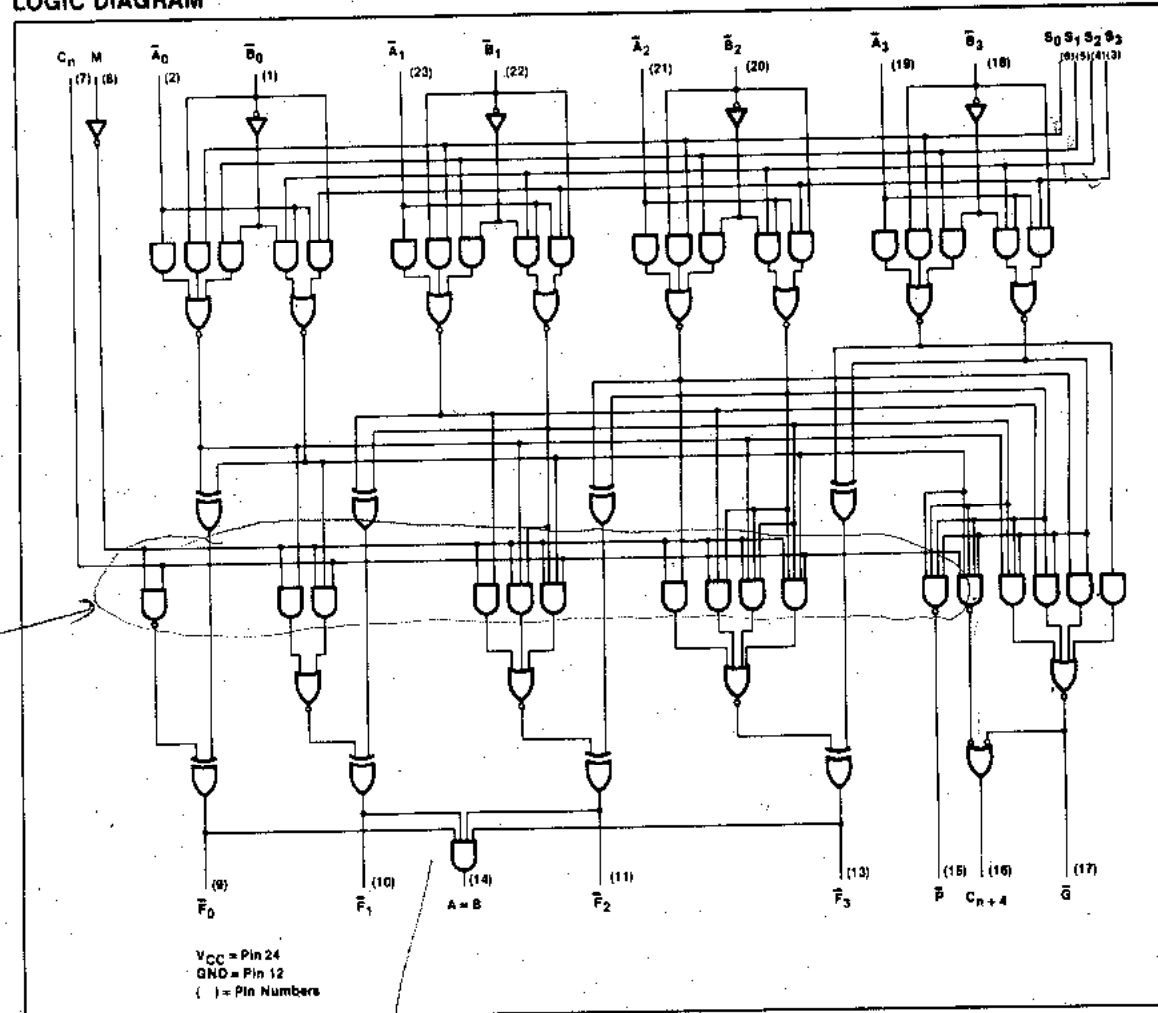
Avec seulement des ou-exclusifs :



Une UAL

SN 74181 (Texas Instruments) : 24 ns

LOGIC DIAGRAM



Choix de l'opération effectuée

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	\bar{A}	A
L	L	L	H	A + B	A + B
L	L	H	L	$\bar{A}\bar{B}$	A + \bar{B}
L	L	H	H	Logical 0	minus 1
L	H	L	L	\overline{AB}	A plus $\bar{A}\bar{B}$
L	H	L	H	\bar{B}	(A + B) plus $\bar{A}\bar{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	\overline{AB}	AB minus 1
H	L	L	L	$\bar{A} + B$	A plus AB
H	L	L	H	$A \oplus B$	A plus B
H	L	H	L	B	(A + \bar{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 + \bar{B}	(A + B) plus A
H	H	H	L	A + B	(A + \bar{B}) plus A
H	H	H	H	A	A minus 1

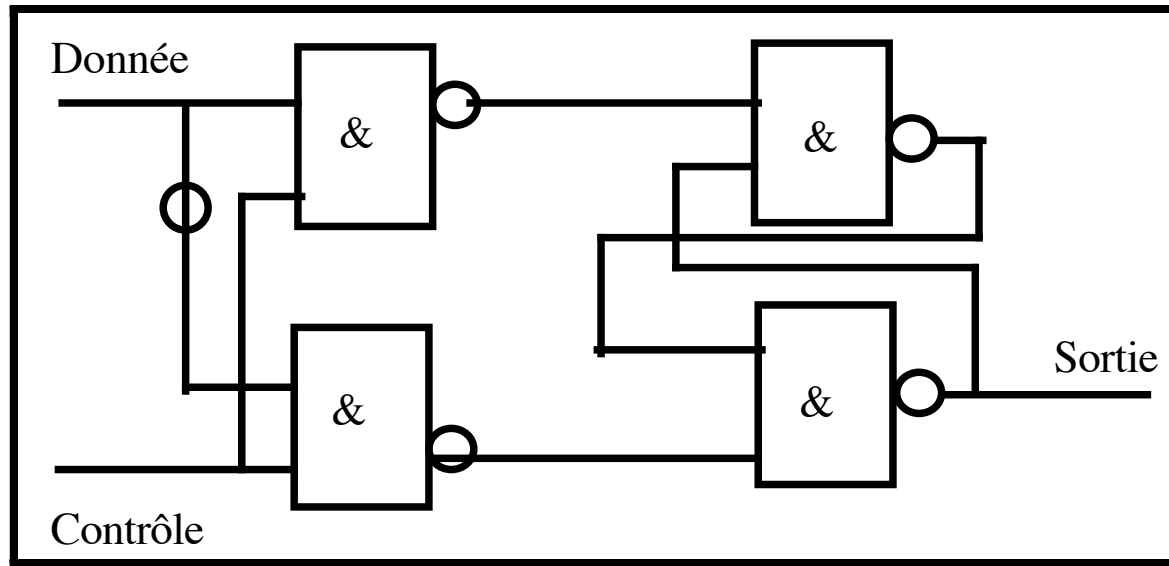
MODE SELECT INPUTS				ACTIVE LOW INPUTS & OUTPUTS	
S ₃	S ₂	S ₁	S ₀	LOGIC (M = H)	ARITHMETIC** (M = L) (C _n = L)
L	L	L	L	\bar{A}	A minus 1
L	L	L	H	\overline{AB}	AB minus 1
L	L	H	L	$\bar{A} + B$	$\bar{A}\bar{B}$ minus 1
L	L	H	H	Logical 1	minus 1
L	H	L	L	$\overline{A + B}$	A plus (A + \bar{B})
L	H	L	H	\bar{B}	AB plus (A + \bar{B})
L	H	H	L	$\overline{A \oplus B}$	A minus B minus 1
L	H	H	H	A + \bar{B}	A + \bar{B}
H	L	L	L	$\bar{A}\bar{B}$	A plus (A + B)
H	L	L	H	A \oplus B	A plus B
H	L	H	L	B	$\bar{A}\bar{B}$ plus (A + B)
H	L	H	H	A + B	A + B
H	H	L	L	Logical 0	A plus A*
H	H	L	H	\overline{AB}	AB plus A
H	H	H	L	AB	$\bar{A}\bar{B}$ plus A
H	H	H	H	A	A

* = LOW voltage

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

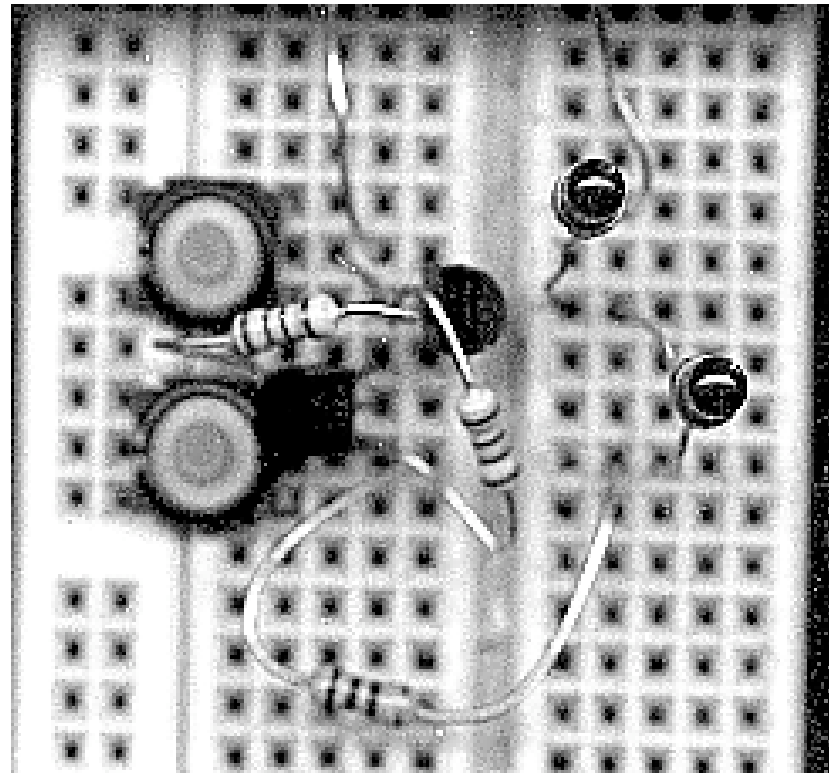
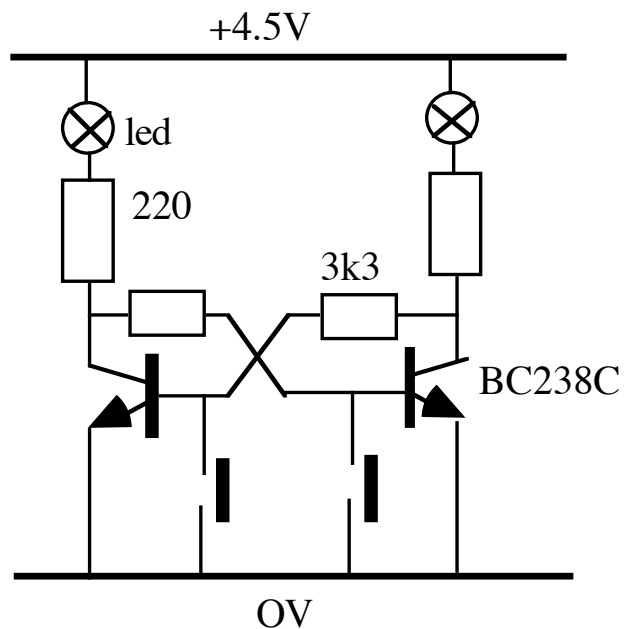
$E(t)$ = Entrée à l'instant t

$S(t)$ = 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 élément puisque elle ne peut être connue à l'avance ? C'est le résultat d'un calcul...

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

Algorithme

- les variables de travail

```

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

```

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)

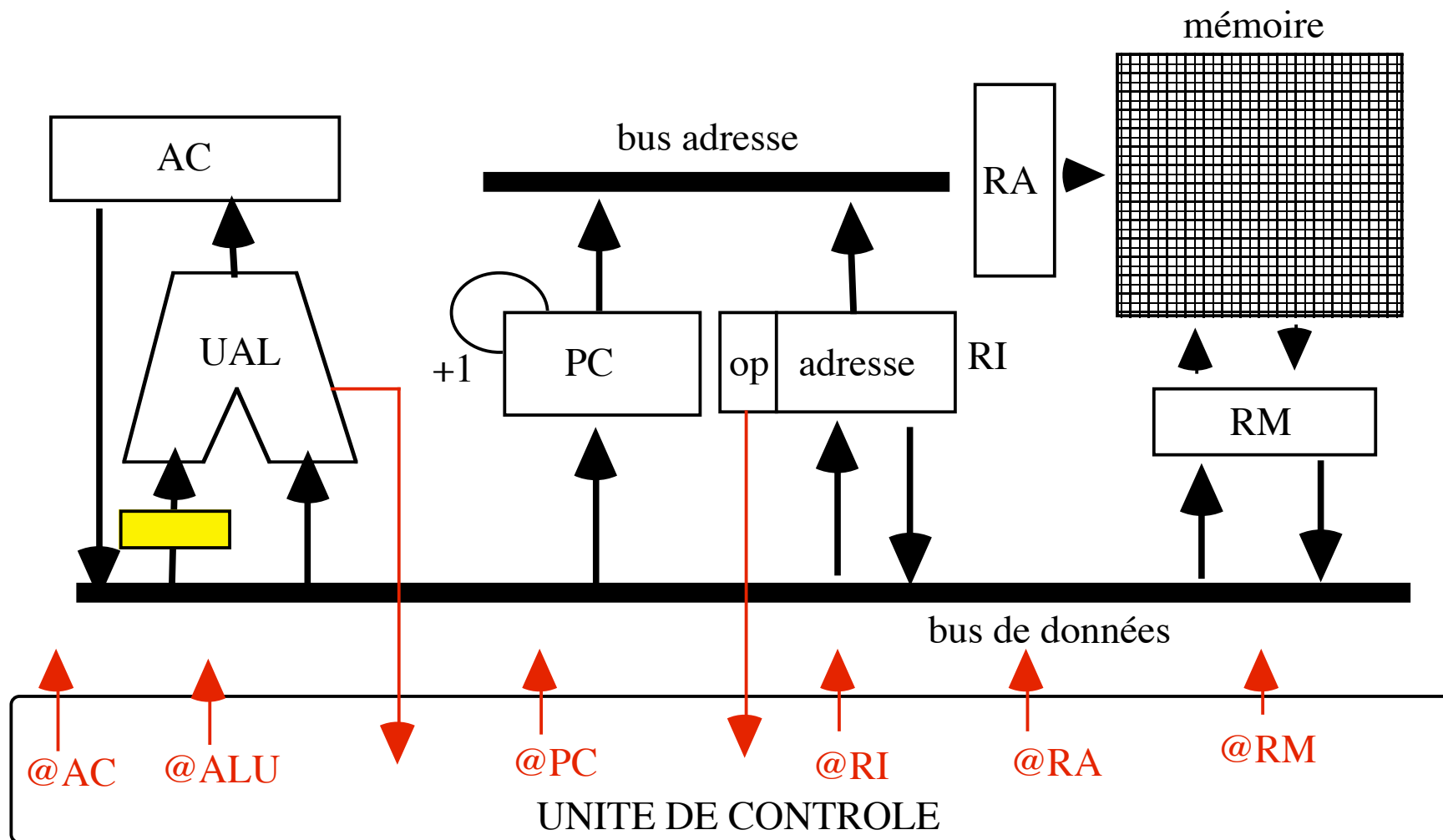
Compteur ordinal, compteur de programme, PC ...

Il faut revoir le codage des instructions pour qu'elles coïncident 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 spécialisées 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 $1.8 \cdot 10^{19}$
- 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) \neq entiers (32 bits) \neq 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 **totallement** 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 (plus 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 baptisée ici 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 gênant : 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