Une petite histoire de l'arithmétique virgule flottante

Jean-Michel Muller
CNRS - Laboratoire LIP
(CNRS-INRIA-ENS Lyon-Université de Lyon)
Journées RAIM – Novembre 2013

http://perso.ens-lyon.fr/jean-michel.muller/

Arithmétique virgule flottante

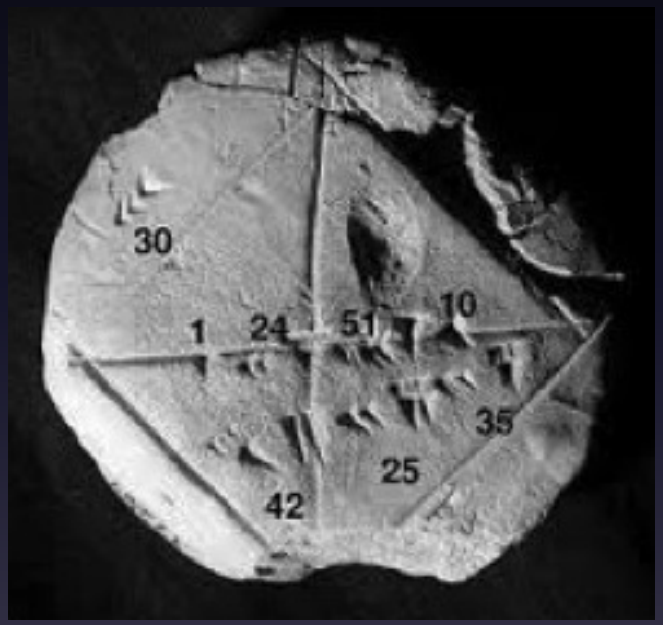
- Notation “scientifique” : *mantisse* et *exposant*

$$x = x_0.x_1x_2\cdots x_p \times \beta^{e_x}$$

- avantages :
  - *dynamique* : représenter de très petits et de très grands nombres de manière compacte ;
  - algorithmes arithmétiques *simples.*

Les Mésopotamiens inventent les mantisses. . .

- Système de base 60 ;
- pas de zéro “à la fin” : on manipule des *mantisses* avec un exposant implicite.



$$1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 1.41421296\cdots$$

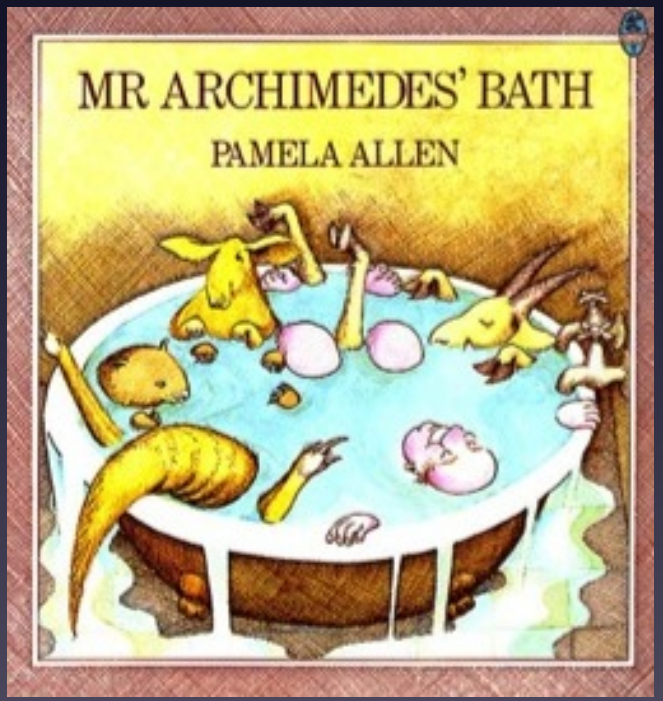
$$\sqrt{2} = 1.414213562\cdots$$

. . . et Archimède invente les exposants

- Traité l'*Arénaire* (compteur de sable : *arena* = sable en Latin) ;
- voulait compter le nombre de grains de sable qui pourraient remplir l'Univers ;
- invention d'une notation exponentielle pour représenter les ordres de grandeur.

Hypothèse :

$$\frac{\text{diamètre Univers}}{\text{diamètre orbite terrestre}} = \frac{\text{diamètre orbite terrestre}}{\text{diamètre Terre}}$$



. . . et Archimède invente les exposants

- point de départ : savait compter en Grec jusqu'à 10^8 (une myriade de myriades – $\mu\upsilon\rho\iota\alpha\varsigma = 10000$) ;
- nombres de la première période :
  - nombres "premiers" : $1 \rightarrow 10^8$;
  - nombres "seconds" : de la forme $10^8 \times$ nombre "premier" ;
  - nombres "troisièmes" : de la forme $10^8 \times$ nombre "second" ;
  - . . .
  - jusqu'aux nombres "10^8èmes" $\rightarrow \Omega = 10^{8 \cdot 10^8}$;
- nombres de la deuxième période : $\Omega \times$ nombres de la 1ère période.

Réponse d'Archimède : on fait tenir au plus 10^{63} grains de sable dans l'Univers.

Notre univers actuel : 10^{80} atomes.

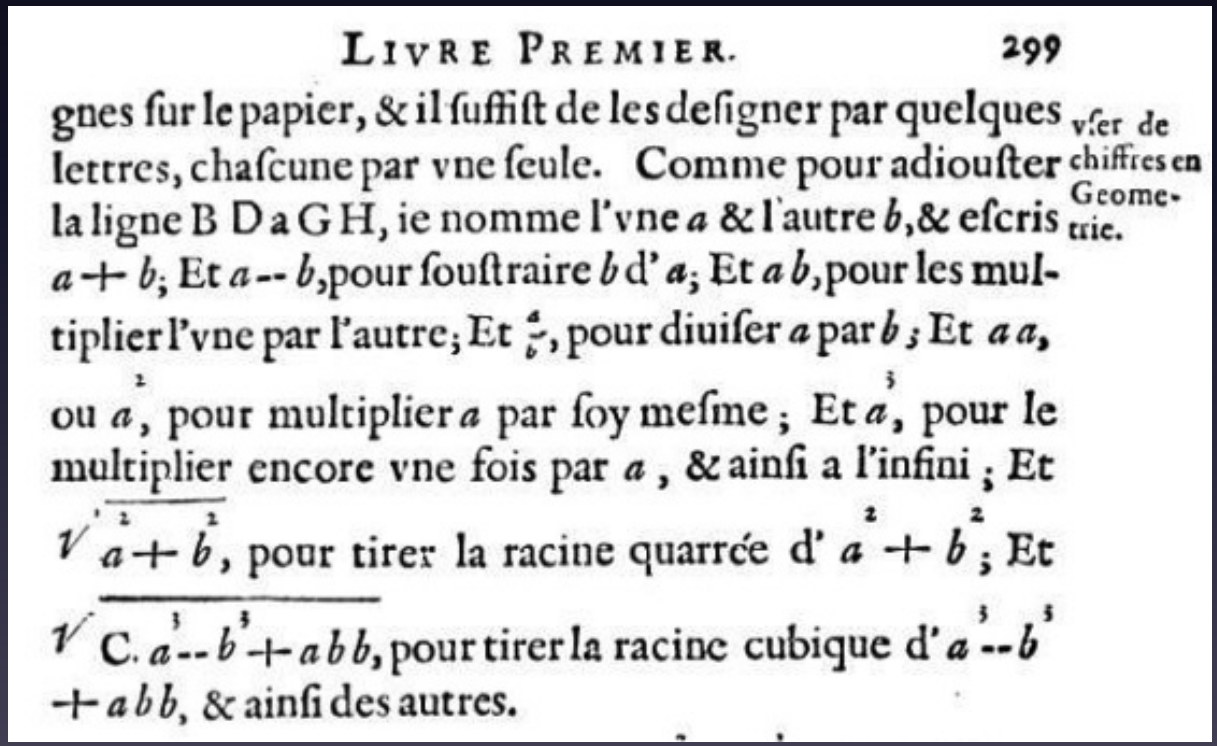
On continue avec la règle à calculs

- Gunter (1624) – 10 ans après que Neper ait fait connaître son invention des logarithmes. Echelle fixe : distances reportées à l'aide d'un compas.

- Règles glissantes :Wingate (1627), cercles concentriques : Oughtred (1630) – probable inventeur du symbole $\times$

La “notation scientifique” des réels

Première étape : notation a^n pour $a \times a \times \cdots \times a$ – Descartes, dans *La Géométrie* (il y invente aussi le symbole $\sqrt{\cdot}$). 1637 ?



LIVRE PREMIER. 299

gnes ſur le papier, & il ſuffiſt de les deſigner par quelques lettres, chaſcune par vne ſeule. Comme pour adiouſter la ligne B D a G H, ie nomme l'vne a & l'autre b, & eſcris $a + b$; Et $a - b$, pour ſouſtraire b d'a; Et ab, pour les multiplier l'vne par l'autre; Et $\frac{a}{b}$, pour diuiſer a par b; Et aa, ou a^2, pour multiplier a par ſoy meſme; Et a^3, pour le multiplier encore vne fois par a, & ainſi a l'infini; Et $\sqrt{a^2 + b^2}$, pour tirer la racine quarrée d'$a^2 + b^2$; Et $\sqrt{C.\ a^3 - b^3 + abb}$, pour tirer la racine cubique d'$a^3 - b^3 + abb$, & ainſi des autres.

vſer de chiffres en Geometrie.

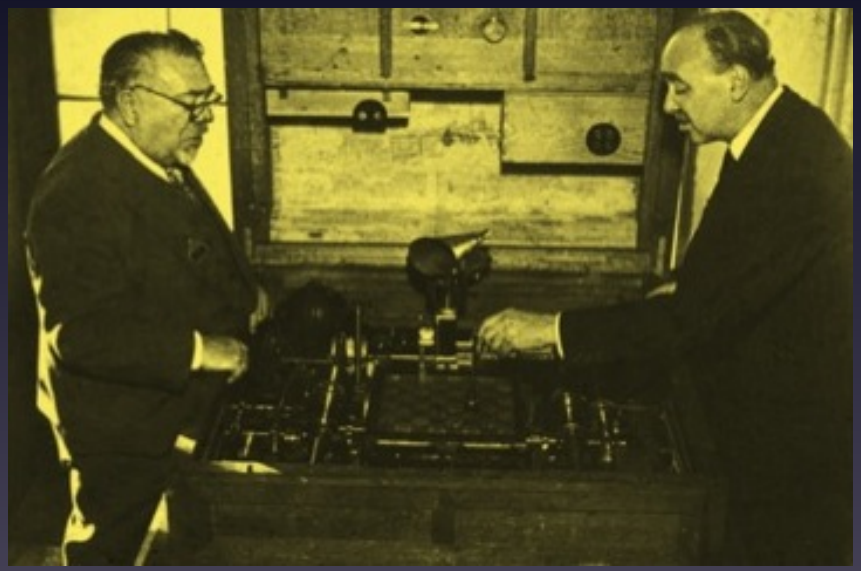
La “notation scientifique” des réels

- à cause de ceci on attribue parfois l’invention de la “notation scientifique” à Descartes ;
- Wallis (1665) puis Newton (1669) : exposants négatifs, rationnels ;
- la notation x^n permet d’écrire un nombre sous la forme $m \times 10^e$, mais cette représentation ne se généralise vraiment qu’au 19ème siècle.

Le net est parfois amusant : sur un site américain la notation scientifique a été *inventée par Descartes puis améliorée par Archimède.*

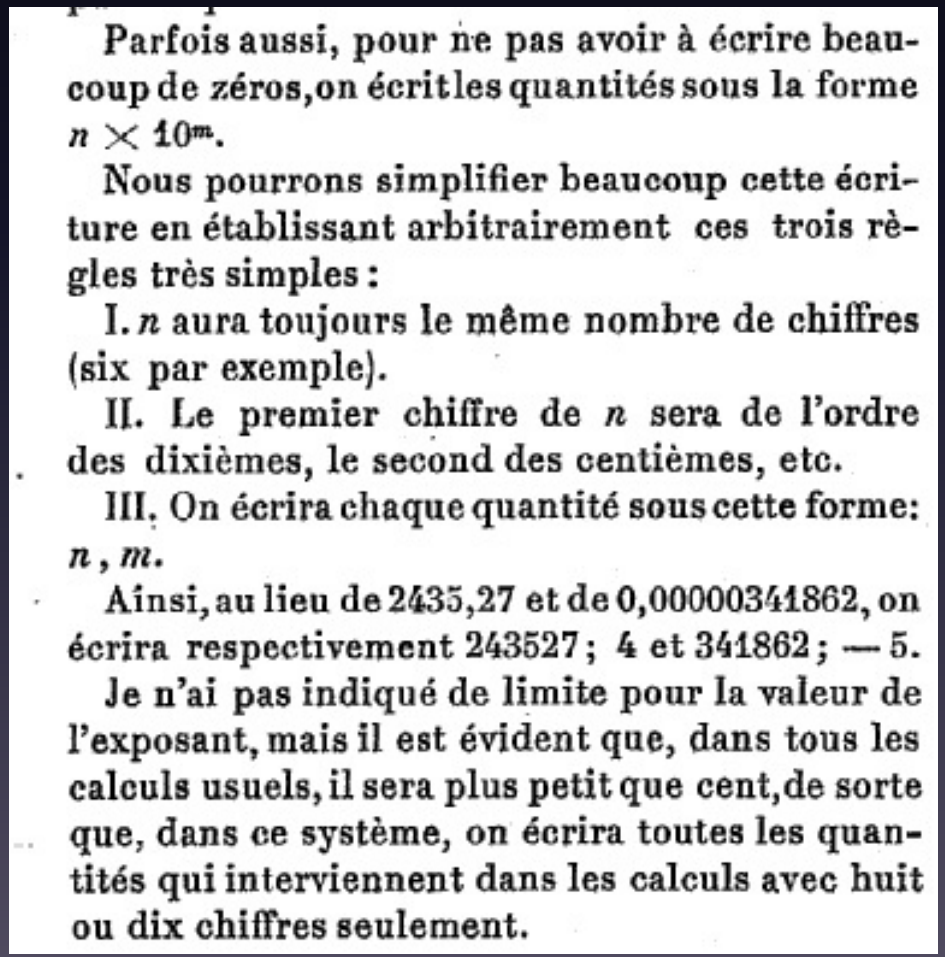
Leonardo Torres y Quevedo (1852–1936)

- version électromécanique (à relais) de la machine analytique de Babbage ;
- première proposition d'une arithmétique virgule flottante (1914) ;
- premier automate joueur d'échec (Roi et Tour contre Roi).



L'article de Torres "Essais sur l'Automatique"

- En Français, *Revue Générale des Sciences*, 15 novembre 1915 ;
- propose de nombreux opérateurs arithmétiques, et contient le paragraphe :



Parfois aussi, pour ne pas avoir à écrire beaucoup de zéros, on écrit les quantités sous la forme $n \times 10^m$.

Nous pourrons simplifier beaucoup cette écriture en établissant arbitrairement ces trois règles très simples :

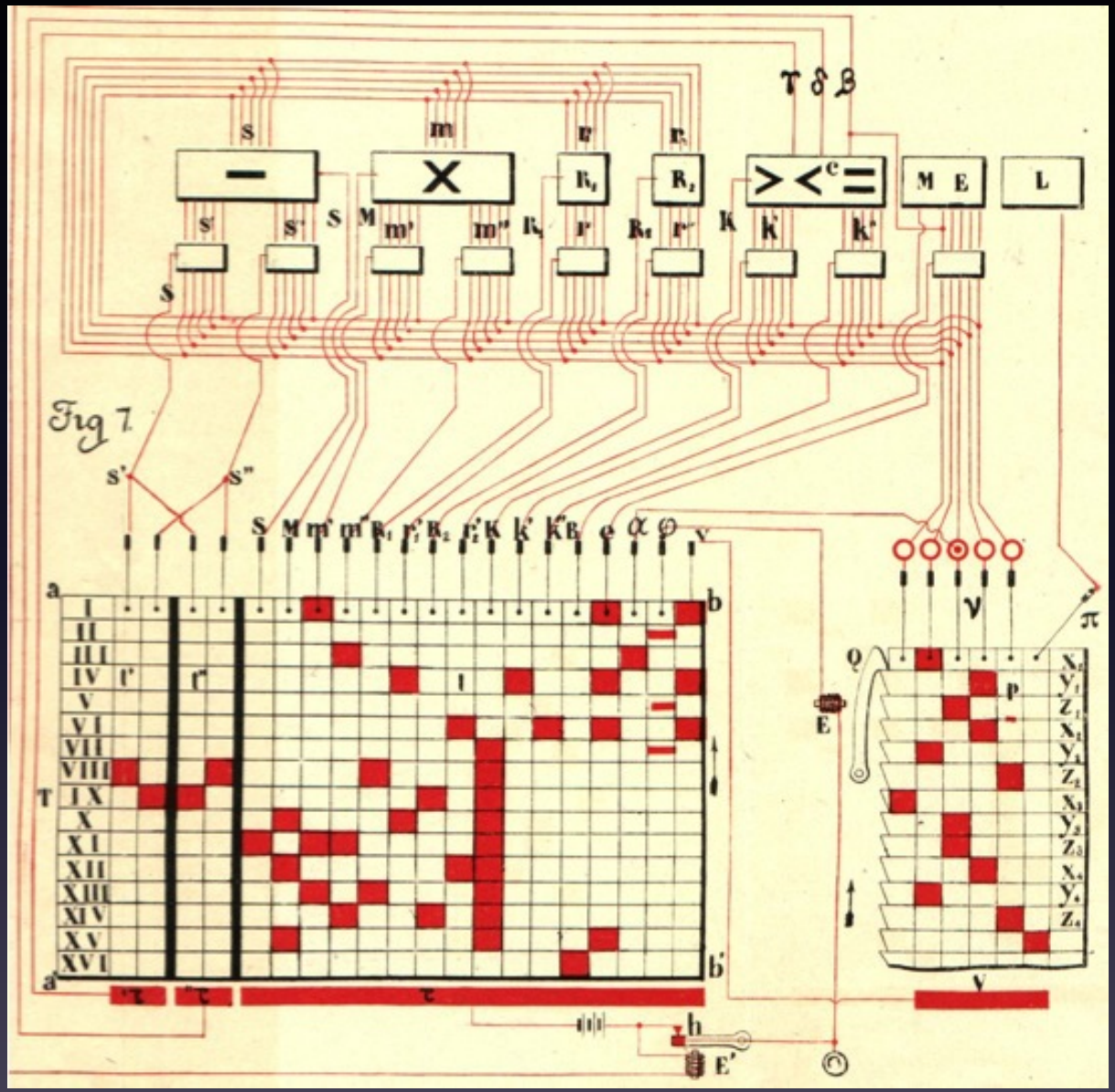
I. n aura toujours le même nombre de chiffres (six par exemple).

II. Le premier chiffre de n sera de l'ordre des dixièmes, le second des centièmes, etc.

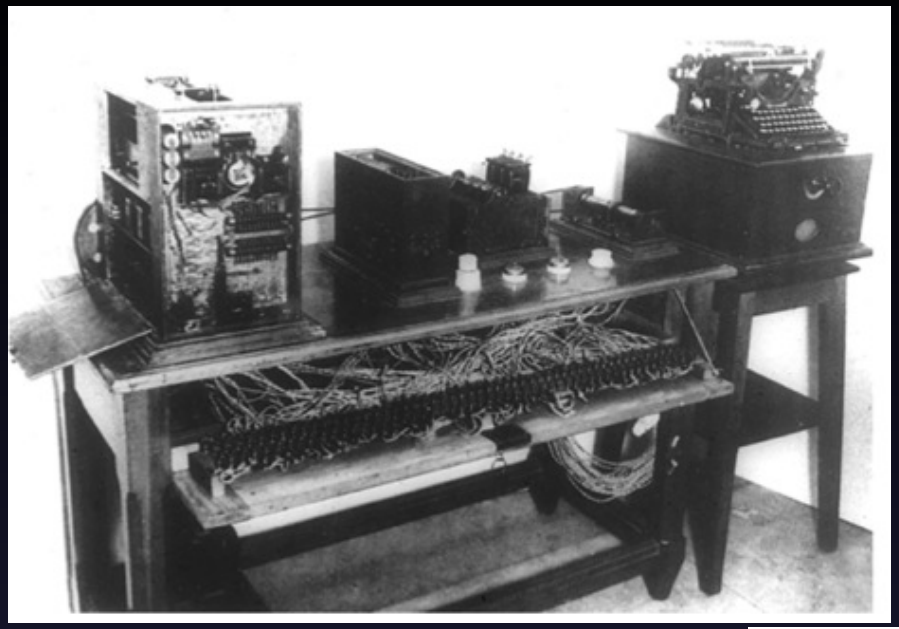
III. On écrira chaque quantité sous cette forme: n, m.

Ainsi, au lieu de 2435,27 et de 0,00000341862, on écrira respectivement 243527; 4 et 341862; — 5.

Je n'ai pas indiqué de limite pour la valeur de l'exposant, mais il est évident que, dans tous les calculs usuels, il sera plus petit que cent, de sorte que, dans ce système, on écrira toutes les quantités qui interviennent dans les calculs avec huit ou dix chiffres seulement.

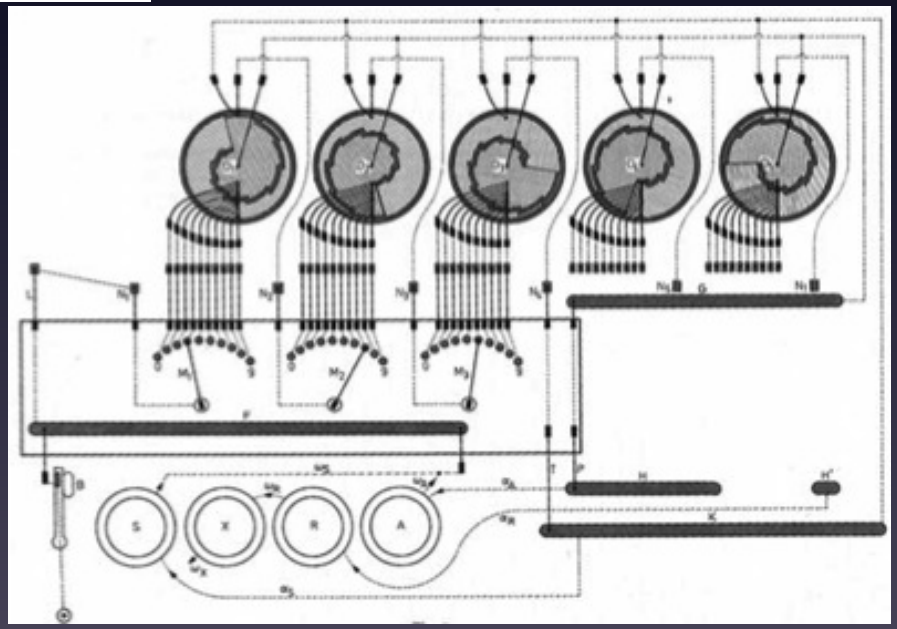


Automate de calcul de $a \cdot (y - z)^2$.



Arithmomètre (1920)

Mécanisme de division



Téléphérique des chutes du Niagara (1916).

Telekino : 1ère (ou 2ème) machine radio-commandée (1901).

Konrad Zuse (1910–1995) : le principal inventeur de l'Ordinateur ?

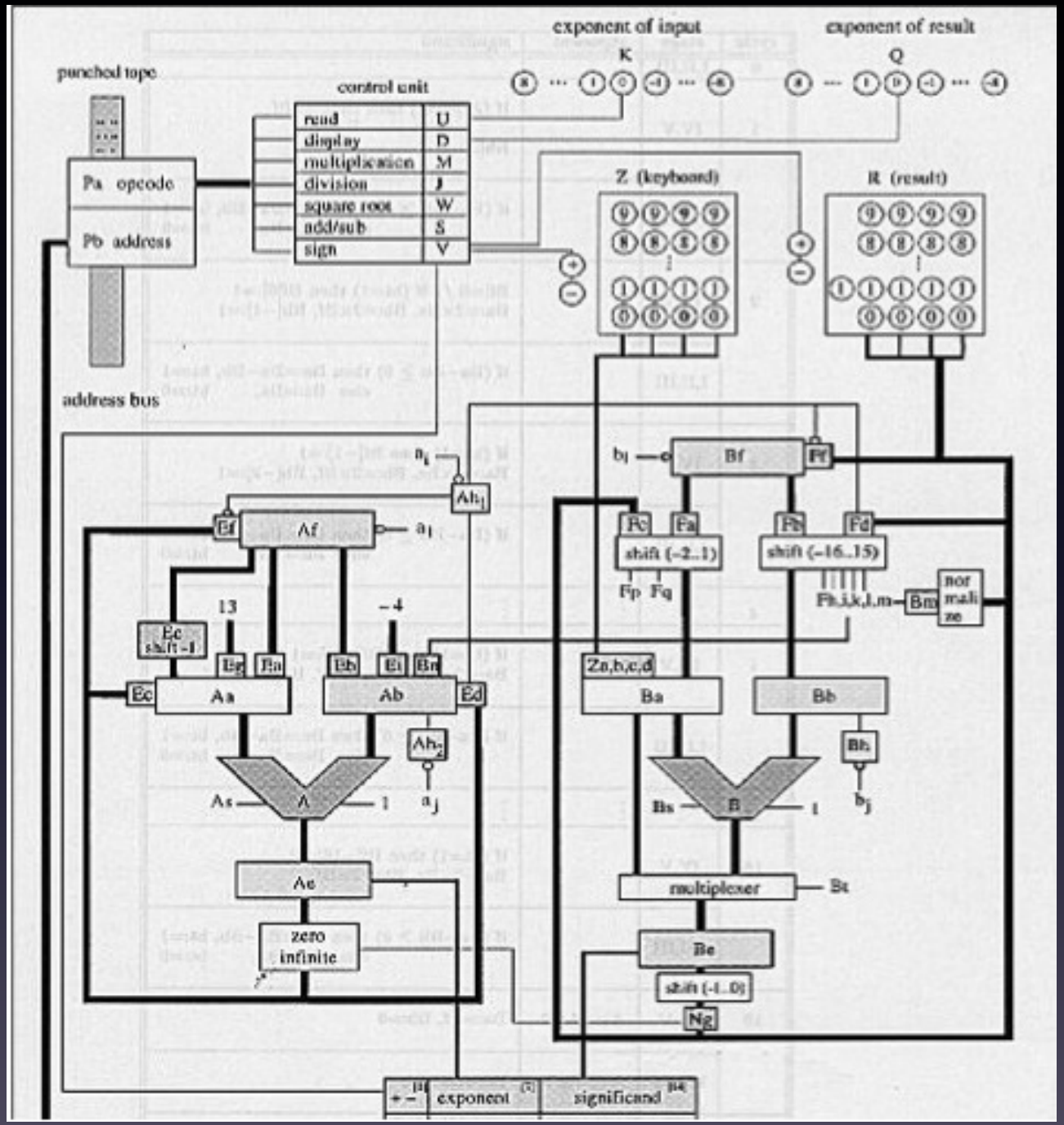
Zuse posant devant une reconstruction du Z3

Les machines Z1 et Z2 de Zuse

- dès le début : système binaire ;
- Z1 (1936–1938) : calculateur mécanique utilisant un moteur électrique ;
  - n'a jamais bien fonctionné ;
  - détruit dans un bombardement.
- Z2 (1938) : arithmétique virgule fixe à l'aide de relais électromécaniques (des relais de téléphone d'occasion) et mémoire mécanique ;
- le Z2 n'était pas fiable mais a suffi comme "preuve de concept" :
  - à convaincre Zuse qu'un calculateur d'envergure était réalisable ;
  - à convaincre le DVL (institut allemand d'aéronautique) de financer ses travaux.

Le Z3 (1941)

- apport décisif de Schreyer (doctorat, 1941) : circuit commutateur bistable ;
- arithmétique virgule flottante ;
- base 2, nombres sur 22 bits :
  - mantisses de 14 bits ;
  - exposants de 7 bits ;
  - 1 bit de signe ;
- représentations spéciales pour $\pm\infty$ et résultats indéterminés, plus de 40 ans avant IEEE 754 ;
- contrairement aux Z1 et Z2, a été complètement opérationnel ;
- Zuse ne l'a pas conçu dans cette optique, mais le Z3 était un calculateur universel.



Architecture du Z3

Quelques autres réalisations de Zuse

- premier langage de haut niveau : le Plankalkül (1942–1946) ;

```
P1 max3 (V0[:8.0], V1[:8.0], V2[:8.0]) => R0[:8.0]
max(V0[:8.0], V1[:8.0]) => Z1[:8.0]
max(Z1[:8.0], V2[:8.0]) => R0[:8.0]
END
P2 max (V0[:8.0], V1[:8.0]) => R0[:8.0]
V0[:8.0] => Z1[:8.0]
(Z1[:8.0] < V1[:8.0]) ? V1[:8.0] => Z1[:8.0]
Z1[:8.0] => R0[:8.0]
END
```

  L'université libre de Berlin a écrit un compilateur en 2000 ;

- Calculateurs S1 et S2 : aérodynamique de bombes à guidage (précurseurs des V1). Probablement récupérés par l'URSS en 1945
  ($\longrightarrow$ 1ers missiles russes ?).

Zuse était aussi un peintre

Von Neumann n'y croyait pas

Dans le célèbre papier de Burks, Goldstine & Von Neumann (1946) :

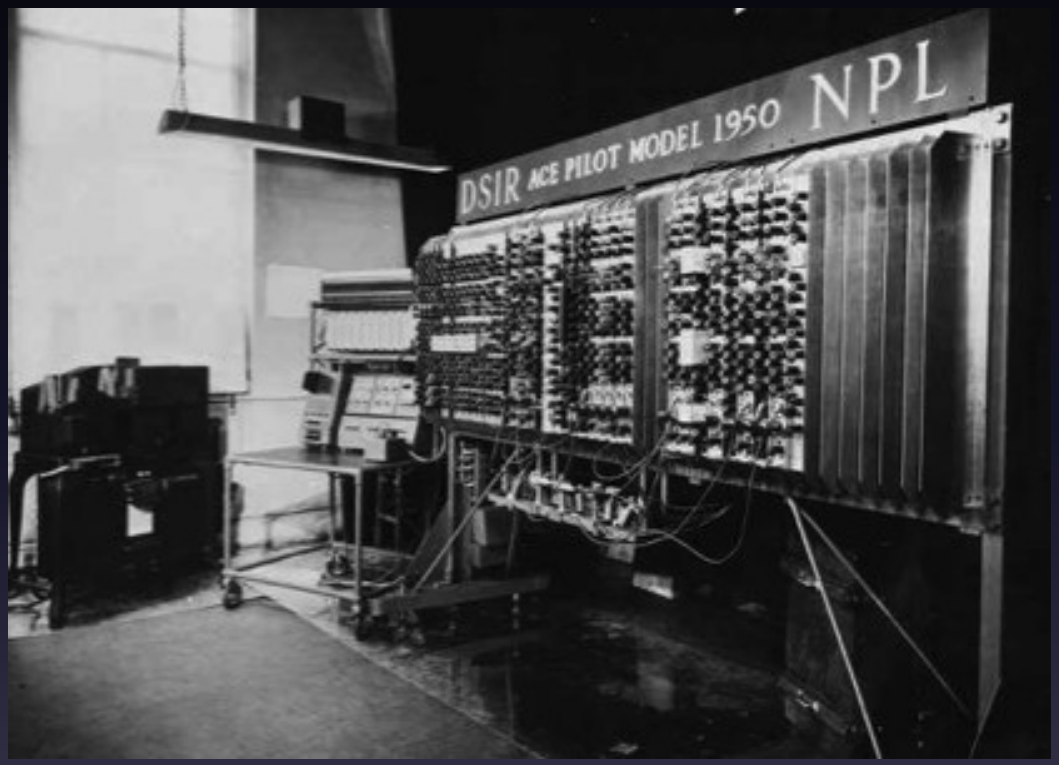
- la VF ajoute de la complexité au calculateur ;
- gérer des facteurs d'échelle "à la main" représente un surcoût en temps négligeable devant le temps de programmation.

Et un peu plus tard (1947) :

> *On ne peut espérer résoudre des systèmes linéaires* $n \times n$*, pour* $n > 15$ *avec une arithmétique VF à* $d = 8$ *chiffres décimaux (même chose pour* $n > 40$ *et* $d = 40$*).*

Il a fallu attendre Wilkinson pour que les techniques d'analyse d'erreur en VF voient le jour.

Mais Turing y croyait



Le *Pilot ACE*, dont le 1er programme a tourné en mai 1950

- ACE : *Automatic Computing Engine. . .* l'autre machine de Turing ;
- National Physics Laboratory, 1946 ;
- Turing embauche Wilkinson comme assistant ;
- Turing quitte le projet en 1947, Wilkinson en prend le contrôle.

Wilkinson et le Pilot ACE

- programmes VF et programmes *multi-précision* écrits par Alway et W. en 1947,avant même que le Pilot ACE ne fonctionne ;
- VF binaire, 30 bits de mantisse, arrondis corrects ;
- Wilkinson met au point la “backward analysis”.

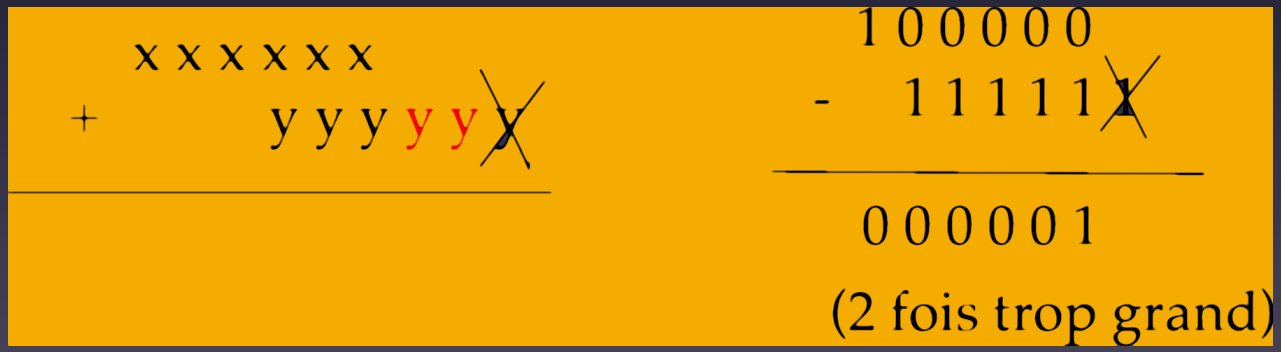
Table 3 Computing speeds of the Pilot ACE, the EDSAC and the Manchester Mark I with floating point operations

Time (ms)	Pilot ACE	EDSAC	Mark I
Add/subtract	8	90	60
Multiply	6	105	80
Divide	34	140	150

Années 50 et 60 : la VF se généralise... mais c'est la pagaille

- Base : 2, 4, 8, 10, 16... des études de Brent et Cody trancheront ;
- seuils d'over/underflow très différents (cf. prochain transparent) ;
- pas la même manière de gérer $1/0$, $0/0$, $\sqrt{-1}$, etc. ;
- spécification floue des opérations : on raisonne en "bits de garde"

Alignement lorsqu'on calcule $x + y$ avec $e_x > e_y$:



`(1 - x)` gagne à être remplacé par `(0.5 - x) + 0.5`

Machine	Underflow λ	Overflow Λ
DEC PDP-11, VAX, F and D formats	$2^{-128} \approx 2.9 \times 10^{-39}$	$2^127 \approx 1.7 \times 10^38$
DEC PDP-10; Honeywell 600, 6000; Univac 110x single; IBM 709X, 704X	$2^{-129} \approx 1.5 \times 10^{-39}$	$2^127 \approx 1.7 \times 10^38$
Burroughs 6X00 single	$8^{-51} \approx 8.8 \times 10^{-47}$	$8^{76} \approx 4.3 \times 10^{68}$
H-P 3000	$2^{-256} \approx 8.6 \times 10^{-78}$	$2^{256} \approx 1.2 \times 10^{77}$
IBM 360, 370; Amdahl1; DG Eclipse M/600; ...	$16^{-65} \approx 5.4 \times 10^{-79}$	$16^{63} \approx 7.2 \times 10^{75}$
Most handheld calculators	10^{-99}	10^{100}
CDC 6X00, 7X00, Cyber	$2^{-976} \approx 1.5 \times 10^{-294}$	$2^{1070} \approx 1.3 \times 10^{322}$
DEC VAX G format; UNIVAC, 110X double	$2^{-1024} \approx 5.6 \times 10^{-309}$	$2^{1023} \approx 9 \times 10^{307}$

Source : Kahan, *Why do we need a Floating-Point Standard*, 1981.

Base 2 vs Base 16...

- Base 16 : 2 bits d'exposants de moins pour même dynamique ;
- on dispose de 14 bits, nombres jusqu'à 4095 $\rightarrow$ exposants base 16 sur 2 bits, et binaires sur 4 bits
- $3889 = F.31 \times 16^2 = 1.111100110001 \times 2^{11}$
  - Hexa : 1111 0011 0001 10
  - Binaire : 1111 0011 00 1011

  $\rightarrow$ c'est la base 16 qui gagne !
- $497.25 = 1.F14 \times 16^2 = 1.111000101 \times 2^7$
  - Hexa : 0001 1111 0001 10
  - Binaire : 1111 0001 01 0111

  $\rightarrow$ c'est la base 2 qui gagne !

Meilleur choix en pire cas ? en cas moyen ?

Le choix de la base

- En pire cas et en cas moyen (distribution de Benford), la base 2 est plus précise que la base 16 ;
- plus généralement, la meilleure base est. . .

Le choix de la base

- En pire cas et en cas moyen (distribution de Benford), la base 2 est plus précise que la base 16 ;
- plus généralement, la meilleure base est. . .

4

Le choix de la base

- En pire cas et en cas moyen (distribution de Benford), la base 2 est plus précise que la base 16 ;
- plus généralement, la meilleure base est. . .

4

sauf si. . .

Le choix de la base

- En pire cas et en cas moyen (distribution de Benford), la base 2 est plus précise que la base 16 ;
- plus généralement, la meilleure base est. . .

4

sauf si. . .

$$1.00110101 \times 2^{e_x} \not\rightarrow 1.00110101 \times 2^{e_x}$$

D’autres exemples (Kahan)

- Quand seule la vitesse compte : sur les Crays, l’overflow était calculé à partir des exposants des entrées, en parallèle avec le calcul effectif du produit

$\longrightarrow$ `1 * x` peut faire un overflow ;

- sur les mêmes, seuls 12 bits de x étaient examinés pour détecter une division par 0 lors du calcul y/x

$\longrightarrow$ `if (x = 0) then z := 17.0 else z := y/x`
peut provoquer une erreur “division par zéro”...
mais comme le multiplieur aussi ne regarde que 12 bits pour décider qu’une opérande est nulle,

```
if (1.0 * x = 0) then z := 17.0 else z := y/x
```

ne pose plus de problème.

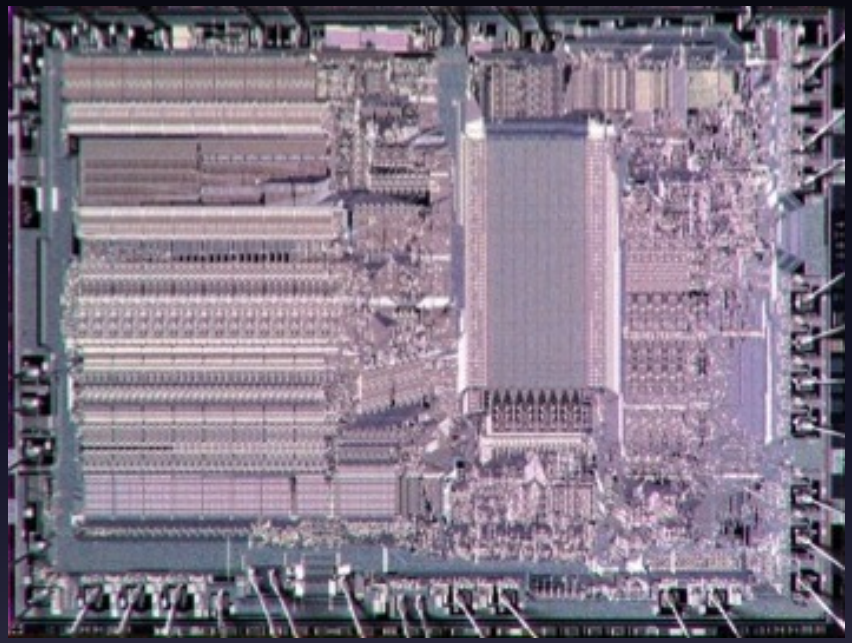
William Kahan

- PhD, Univ. Toronto, 1958 ;
- rencontre Wilkinson lors de son PostDoc en Angleterre ;
- à programmé à peu près toutes les machines de l'époque ;
- a contribué à la conception de la calculatrice HP35 (1972) ;
- arithmétique VF du 8087 ;
- en parallèle (1977), 1ères discussions autour du futur standard IEEE 754.

La HP35 : l'objet qui a "tué" la règle à calcul

Encore beaucoup d'informations à
`http://www.eecs.berkeley.edu/~wkahan`

Le 8087 d'Intel (1980)



- coprocesseur du 8086 ;
- $ 200 environ ;
- fonctions élémentaires en VF (Cordic) ;
- première “presque” implantation de ce qui sera 5 ans + tard IEEE 754 ;
- entrée de la VF rapide et propre dans le monde du personal computing ;
- format “étendu” avec mantisses de 64 bits (+ grande taille qui ne change pas le temps de cycle) ;
- 5 MHz.

intel

8087
MATH COPROCESSOR

- **Adds Arithmetic, Trigonometric, Exponential, and Logarithmic Instructions to the Standard 8086/8088 and 80186/80188 Instruction Set for All Data Types**
- **CPU/8087 Supports 7 Data Types: 16-, 32-, 64-Bit Integers, 32-, 64-, 80-Bit Floating Point, and 18-Digit BCD Operands**
- **Compatible with IEEE Floating Point Standard 754**
- **Available in 5 MHz (8087), 8 MHz (8087-2) and 10 MHz (8087-1): 8 MHz 80186/80188 System Operation Supported with the 8087-1**
- **Adds 8 x 80-Bit Individually Addressable Register Stack to the 8086/8088 and 80186/80188 Architecture**
- **7 Built-In Exception Handling Functions**
- **MULTIBUS System Compatible Interface**

The Intel 8087 Math CoProcessor is an extension to the Intel 8086/8088 microprocessor architecture. When combined with the 8086/8088 microprocessor, the 8087 dramatically increases the processing speed of computer applications which utilize mathematical operations such as CAM, numeric controllers, CAD or graphics.

The 8087 Math CoProcessor adds 68 mnemonics to the 8086 microprocessor instruction set. Specific 8087 math operations include logarithmic, arithmetic, exponential, and trigonometric functions. The 8087 supports integer, floating point and BCD data formats, and fully conforms to the ANSI/IEEE floating point standard.

The 8087 is fabricated with HMOS III technology and packaged in a 40-pin cerdip package.

IEEE 754-1985

- choix de la base 2, de formats (32 bits, 64 bits) ;
- deux idées fortes :
  - système clos : même les opérations “illicites” ($1/0$; $\sqrt{-5}$) fournissent un résultat, qui doit pouvoir être réutilisable en entrée ;
  - arrondi correct : une fonction d’arrondi $\circ$ étant choisie, le calcul en machine de $a \star b$ donne

$$\circ(a \star b)$$

$\rightarrow$ amélioration de la *portabilité*, de la *prouvabilité* et de la *qualité numérique* des programmes.

Erreur de l'addition VF (Møller, Knuth, Dekker)

Premier résultat : représentabilité. $\mathrm{RN}(x) = x$ arrondi au plus près.

Lemme 1

Soient a et b deux nombres VF. Soient

$$s = RN(a + b)$$

et

$$r = (a + b) - s.$$

s'il n'y a pas de dépassement de capacité en calculant s, alors r est un nombre VF.

Obtenir r : l'algorithme fast2sum (Dekker)

Théorème 1 (Fast2Sum (Dekker))

(base ≤ 3) Soient a et b des nombres VF vérifiant $|a| \geq |b|$. Algorithme suivant : s et r t.q.

- $s + r = a + b$ *exactement ;*
- s *est "le" nombre VF le plus proche de* $a + b$.

Algorithme 1 (FastTwoSum)

$s \leftarrow RN(a + b)$
$z \leftarrow RN(s - a)$
$r \leftarrow RN(b - z)$

Programme C 1

```
s = a+b;
z = s-a;
r = b-z;
```

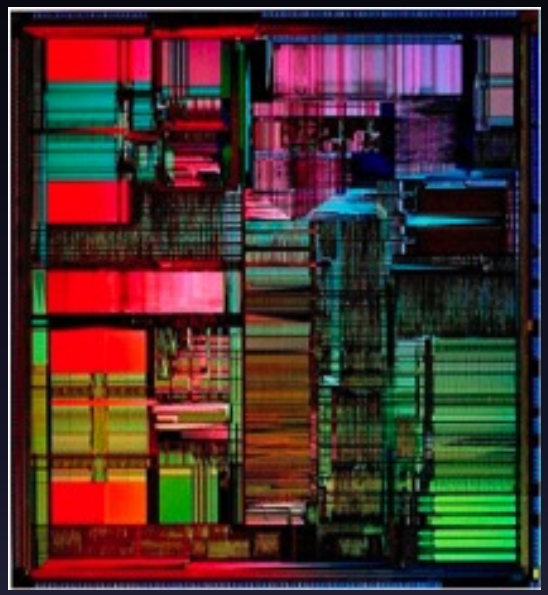
Se méfier des compilateurs "optimisants".

Arithmétique (presque) bien spécifiée

Tout est prêt pour faire de la preuve, sauf qu'à l'époque...

- les ingénieurs/scientifiques n'en éprouvent pas vraiment le besoin : ils font de la simulation tranquilles au sol ;
- les outils de preuve formelle ne sont pas encore prêts ;
- un point souvent négatif : utilisation cachée possible d'une plus grande précision ;
- pour prouver un algorithme, il faut le connaître : culte du secret ;
- il n'y avait pas encore eu de très gros problème ;
- ... et puis chez Intel, Motorola, etc. il y avait à ce moment là un côté "tonton bricoleur" sympathique mais dangereux.

Automne 1994 : la précision d'une règle à calcul



- Thomas Nicely (Lynchburg Univ.) : constante de Brun

$$\left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\cdots$$

  (couples de nombres 1ers jumeaux). Viggo Brun, 1919 : la série converge.

- résultats pas en accord avec les précédents. Dans un tel cas on soupçonne :

  1. le programme ;
  2. le compilateur ;
  3. en dernier recours le processeur.

- le Pentium donnait un résultat incorrect pour 1/824633702441 (824633702441 et 824633702443 sont jumeaux).

Le “bug” du Pentium

- erreur dans l’algorithme de division (SRT de base 4) ;
- nombreux quotients faux. Pire cas : 4195835.0/3145727.0 donne 1.333**73906802** au lieu de 1.3338204491 ;
- tempête électronique sur Internet ;
- Intel a dû remplacer les Pentium défectueux (coût : peut-être 400M$) ;
- la vraie perte a été en termes d’image de marque.

Après ceci : vrai changement de stratégie

- fin du secret sur les algorithmes VF : division de l’Itanium publiée dans les actes d’Arith14 (1999) ;
- preuve formelle : Intel embauche Harrison, AMD embauche Russinoff.

Que sont les Pentium devenus ?

Arithmétique déterministe ?

Programme C :

```
double a = 1848874847.0;
double b = 19954562207.0;
double c;
c = a * b;
printf("c = %20.19e\n", c);
return 0;
```

Selon l'environnement, `3.68934881474191032e+19` ou `3.68934881474191114e+19` (nombre Binary64 le plus proche du résultat exact).

Cause : double arrondi

Nombreux exemples : David Monniaux.

D’autres choses en vrac. . .

- algorithmes Compensés : *Compensated summation* de Kahan, algorithmes de sommation de Pichat, Neumaier, Rump, . . . ; algorithmes compensés pour produits scalaire, évaluation de polynômes, etc.
- arithmétique d’intervalles ;
- perturbation des arrondis : Vignes et Laporte (1974) ;
- arrondi correct des fonctions : dilemme du fabricant de tables (Arénaire, depuis 1997 environ) ;
- coq et flottant (une bonne partie de notre communauté)

Et la suite ?

- nouvelle mouture (2008) de IEEE 754 :
  - prise en compte de l'existant (FMA, quad precision) ;
  - gestion de la co-existence de plusieurs formats ;
  - meilleure insertion de la VF décimale ;
  - nouveautés (recommandation sur fonctions "élémentaires") ;
  - spécification de formats à "grande précision".
- Outils : on ne fera plus d'opérateurs (matériels ou logiciels) mais des générateurs d'opérateurs
- challenge : marier reproductibilité et parallélisme massif.

Quelques sources

- Paul E. Ceruzzi, *A History of Modern Computing*, The MIT Press, 2003 ;
- Brian Randell, *From Analytical Engine to Electronic Digital Computer : The Contributions of Ludgate, Torres and Bush*, Annals of the History of Computing, Vol. 4, oct. 1982 ;
- Paul E. Ceruzzi, *The Early Computers of Konrad Zuse, 1925–1945*, Vol. 3, juillet 1981 ;
- Konrad Zuse internet archive : `http://zuse.zib.de` ;
- Iason Kastanis, The origins and the development of the first high-level programming languages, accessible via la Konrad Zuse internet archive ;
- Jack Copeland, *Alan Turing's Automatic Computing Engine*, Oxford University Press, 2005 ;
- pages Wikipedia l'Arénaire, Leonardo Torres Quevedo, Konrad Zuse ; Intel 8087 ;