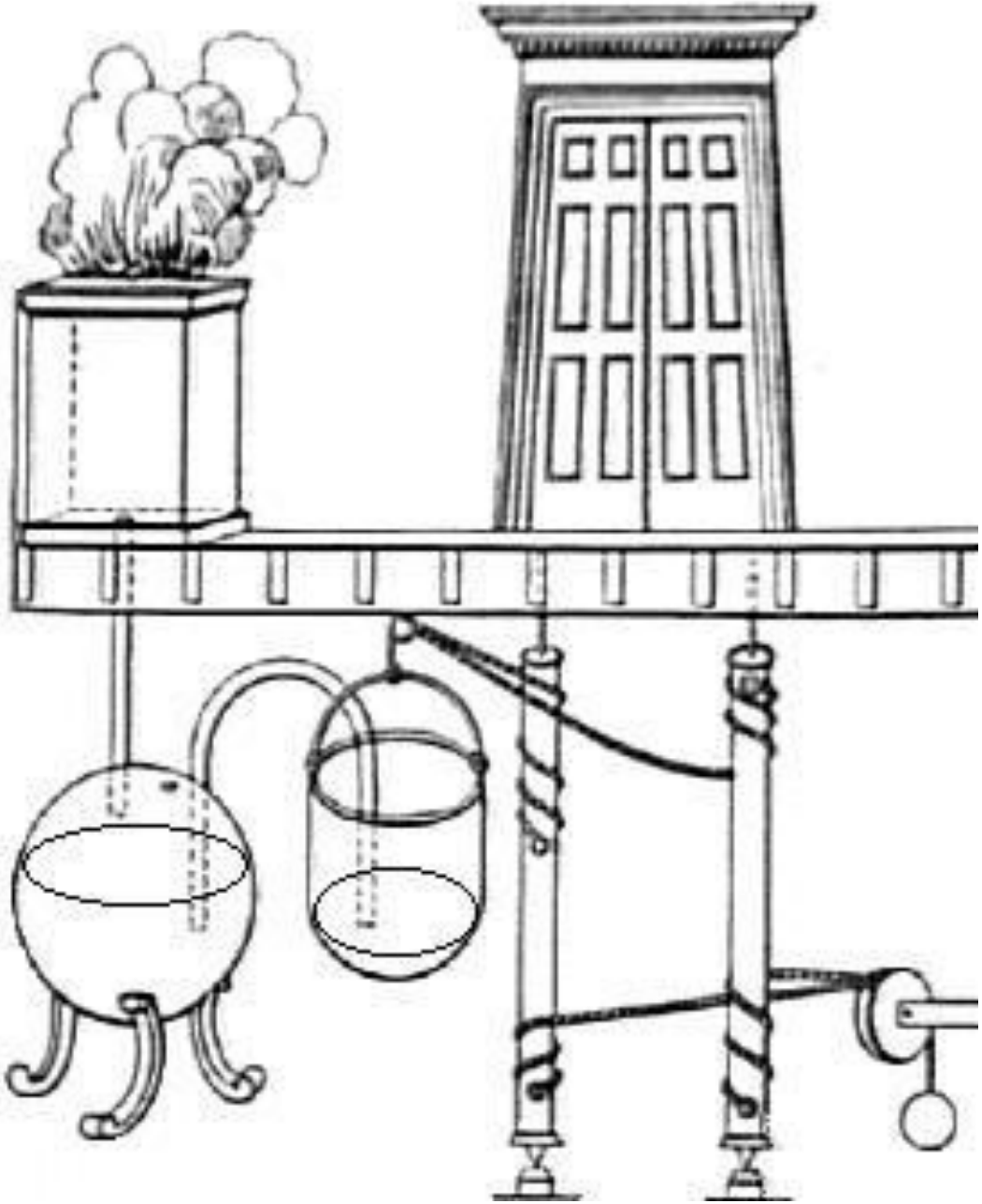
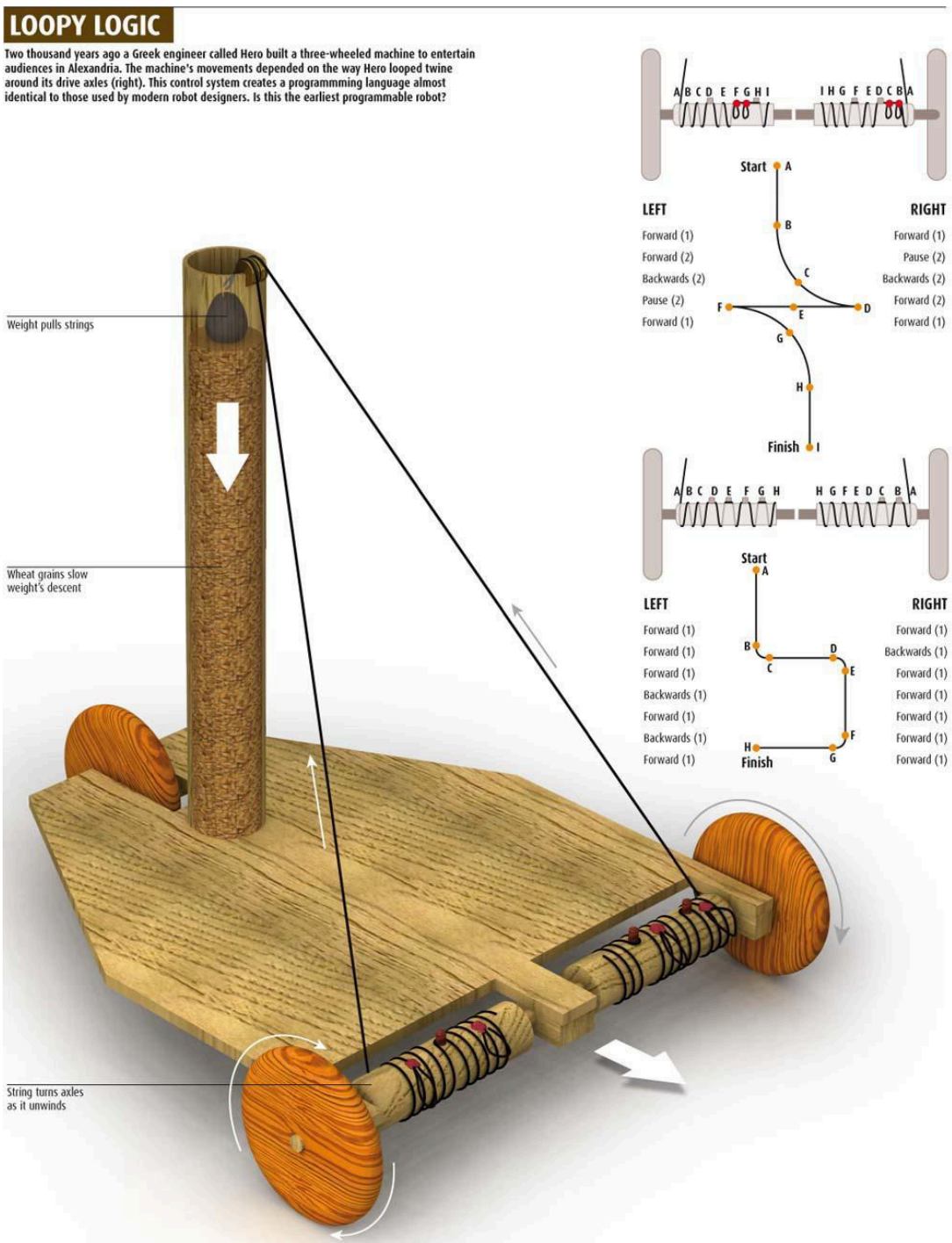
Le origini della programmazione:

le prime macchine
e i linguaggi per programmarle

Lorenzo Repetto

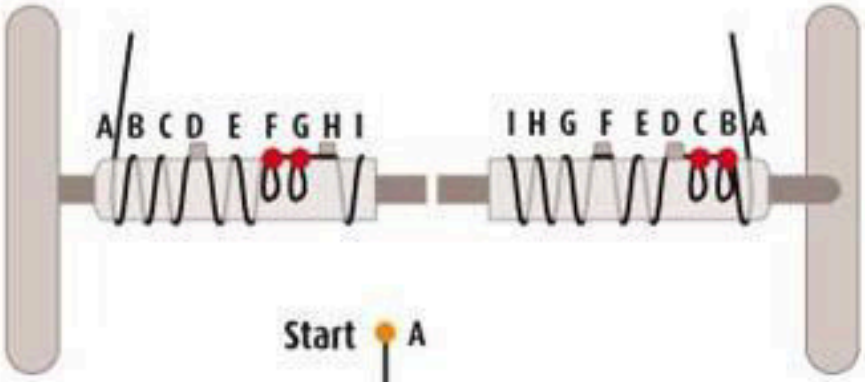
lorenzo.repetto@istruzione.it

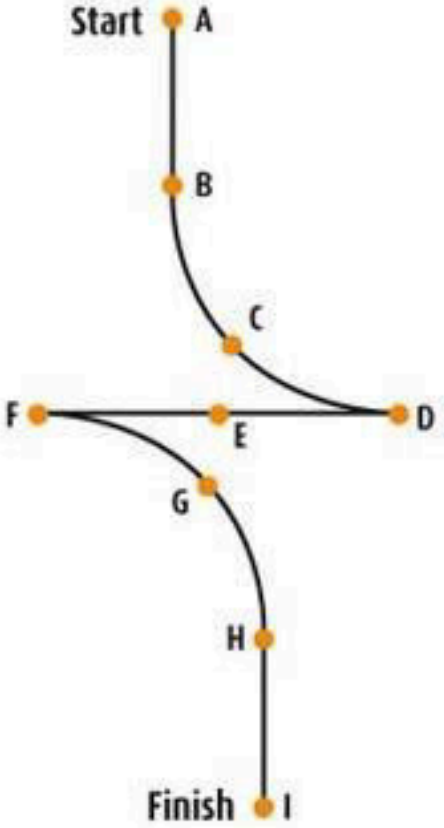




Erone di Alessandria

(circa I secolo d.C.)



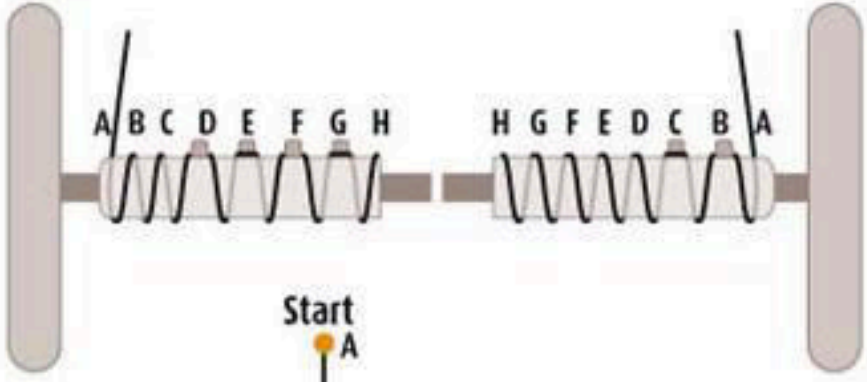


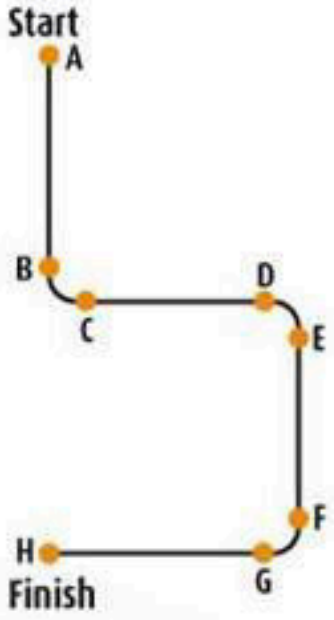
LEFT

Forward (1)
Forward (2)
Backwards (2)
Pause (2)
Forward (1)

RIGHT

Forward (1)
Pause (2)
Backwards (2)
Forward (2)
Forward (1)





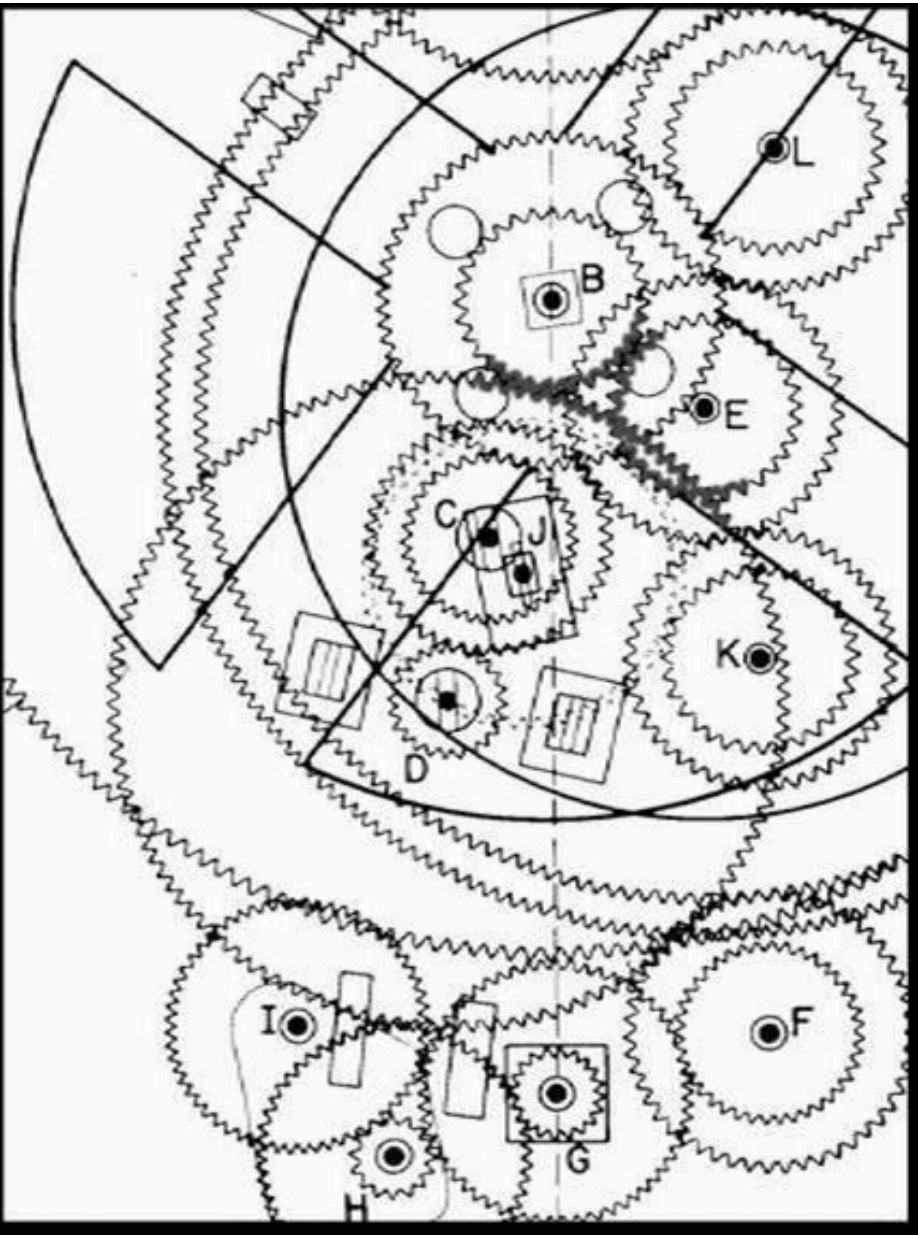
LEFT

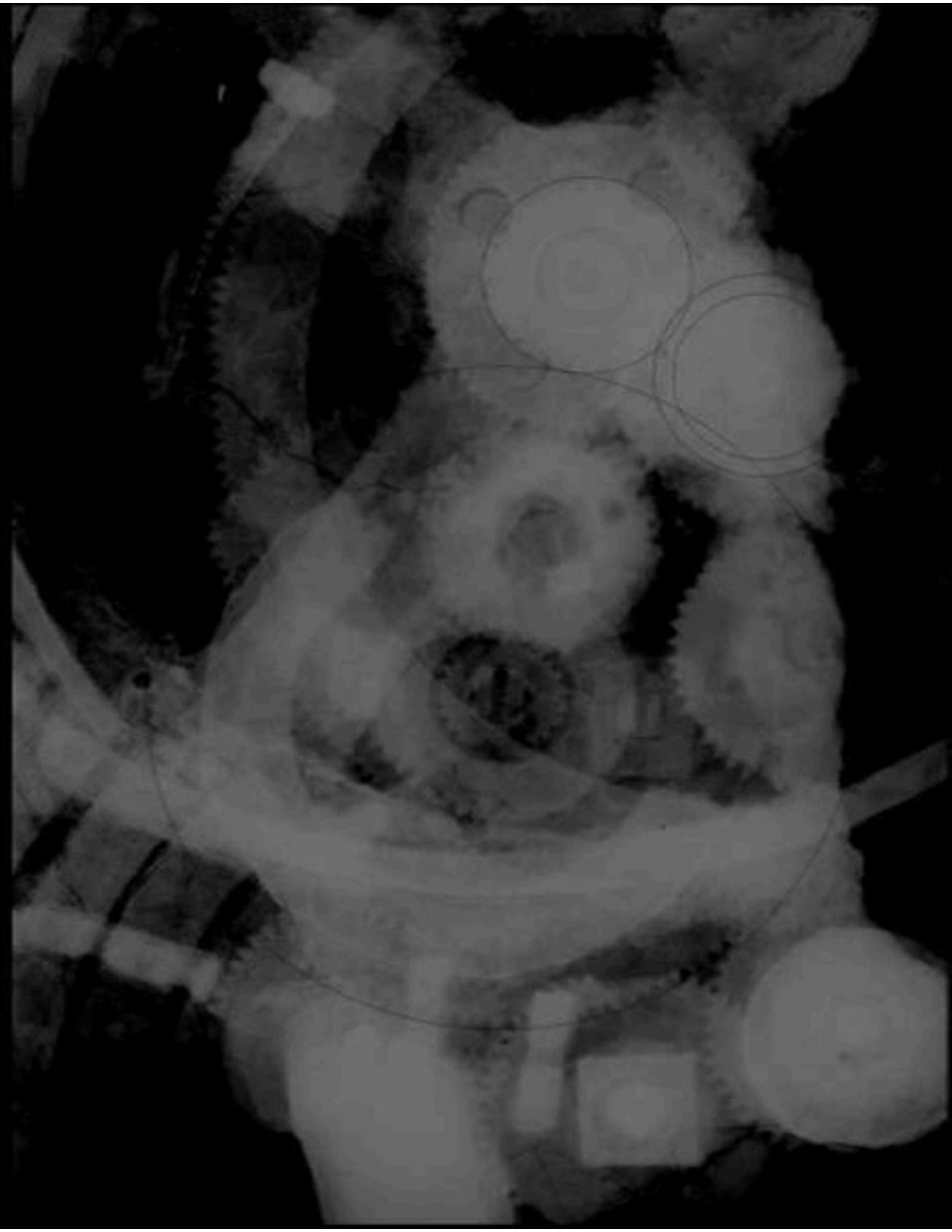
Forward (1)
Forward (1)
Forward (1)
Backwards (1)
Forward (1)
Backwards (1)
Forward (1)

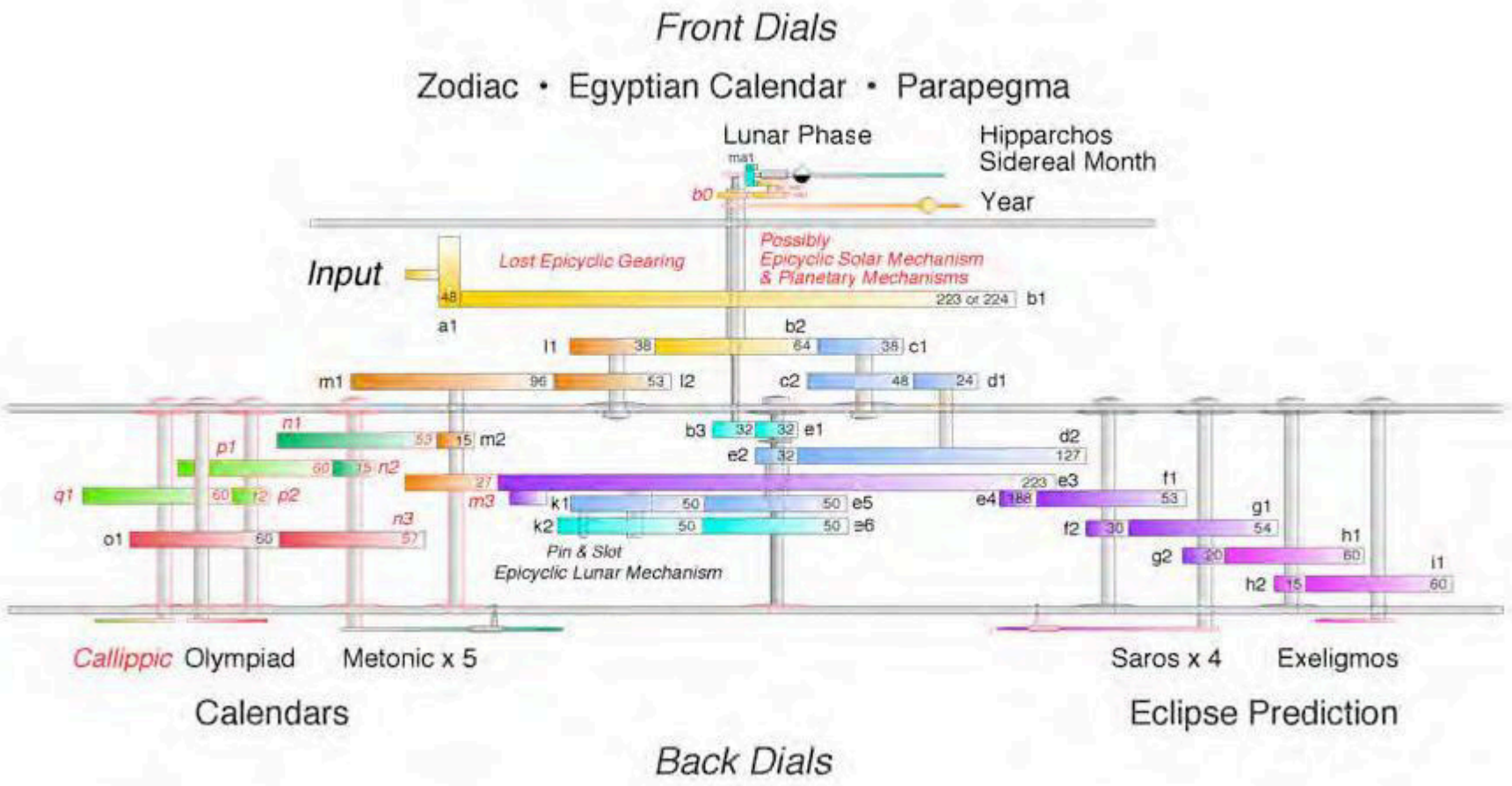
RIGHT

Forward (1)
Backwards (1)
Forward (1)
Forward (1)
Forward (1)
Forward (1)
Forward (1)

Meccanismo di Antikythera (fine II – inizio I sec. a.C.)


L
B
E
C
J
K
D
I
F
G
H

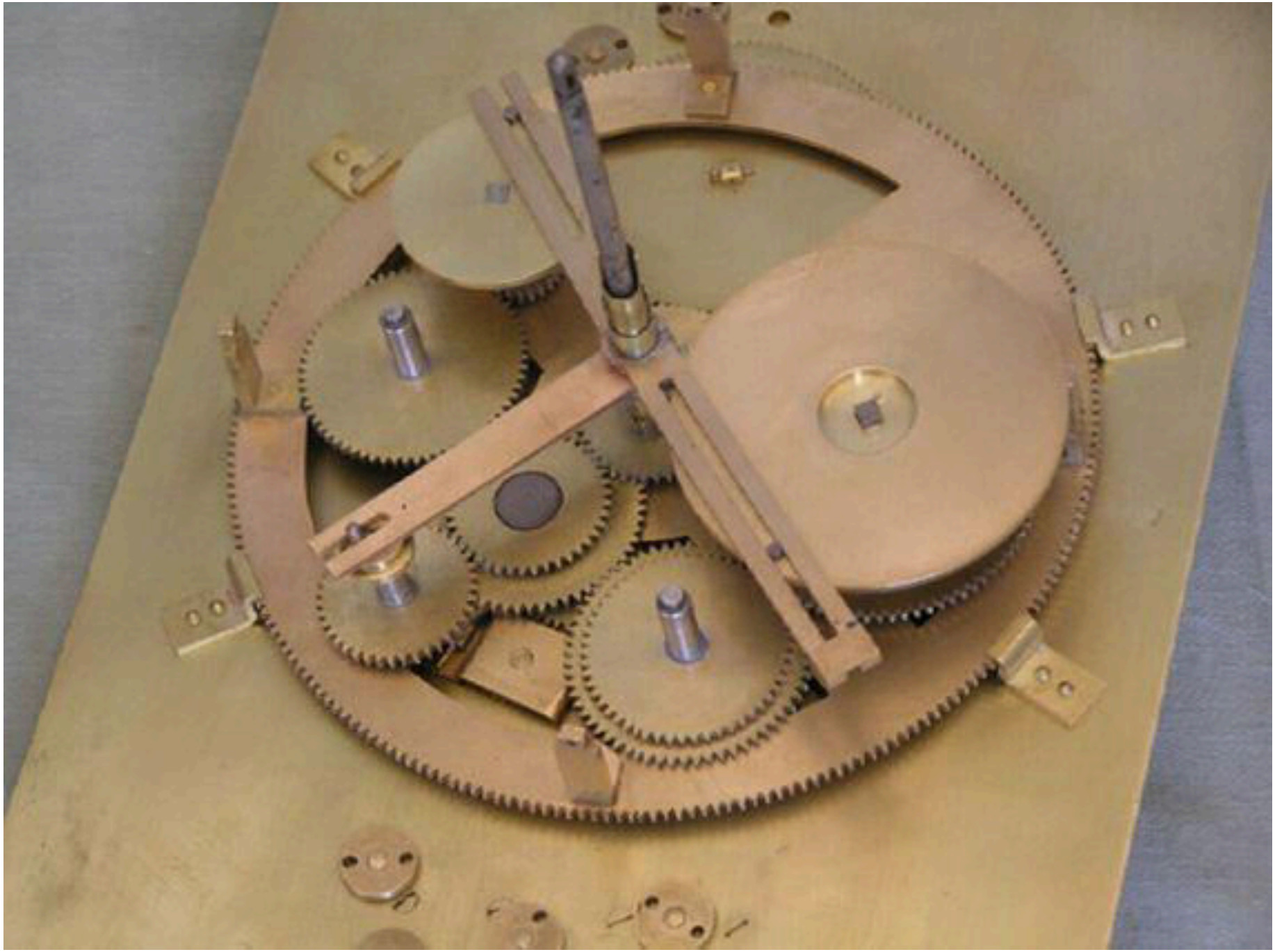




The Antikythera Mechanism Research Project

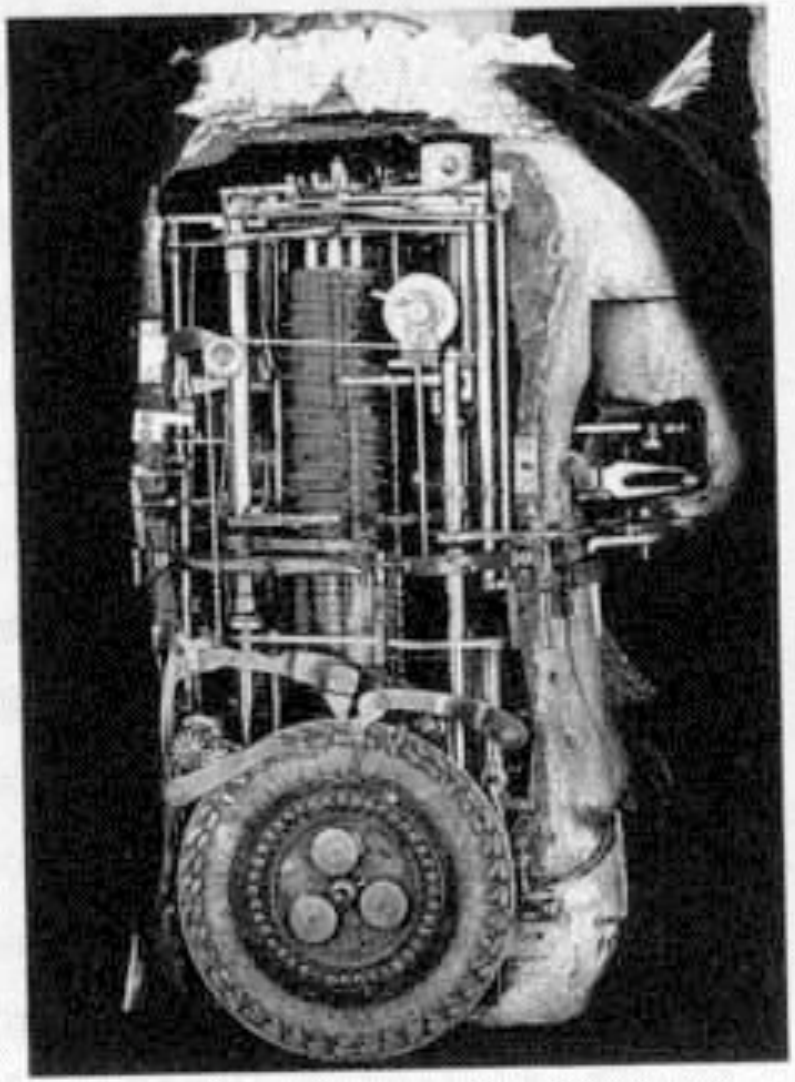
www.antikythera-mechanism.gr

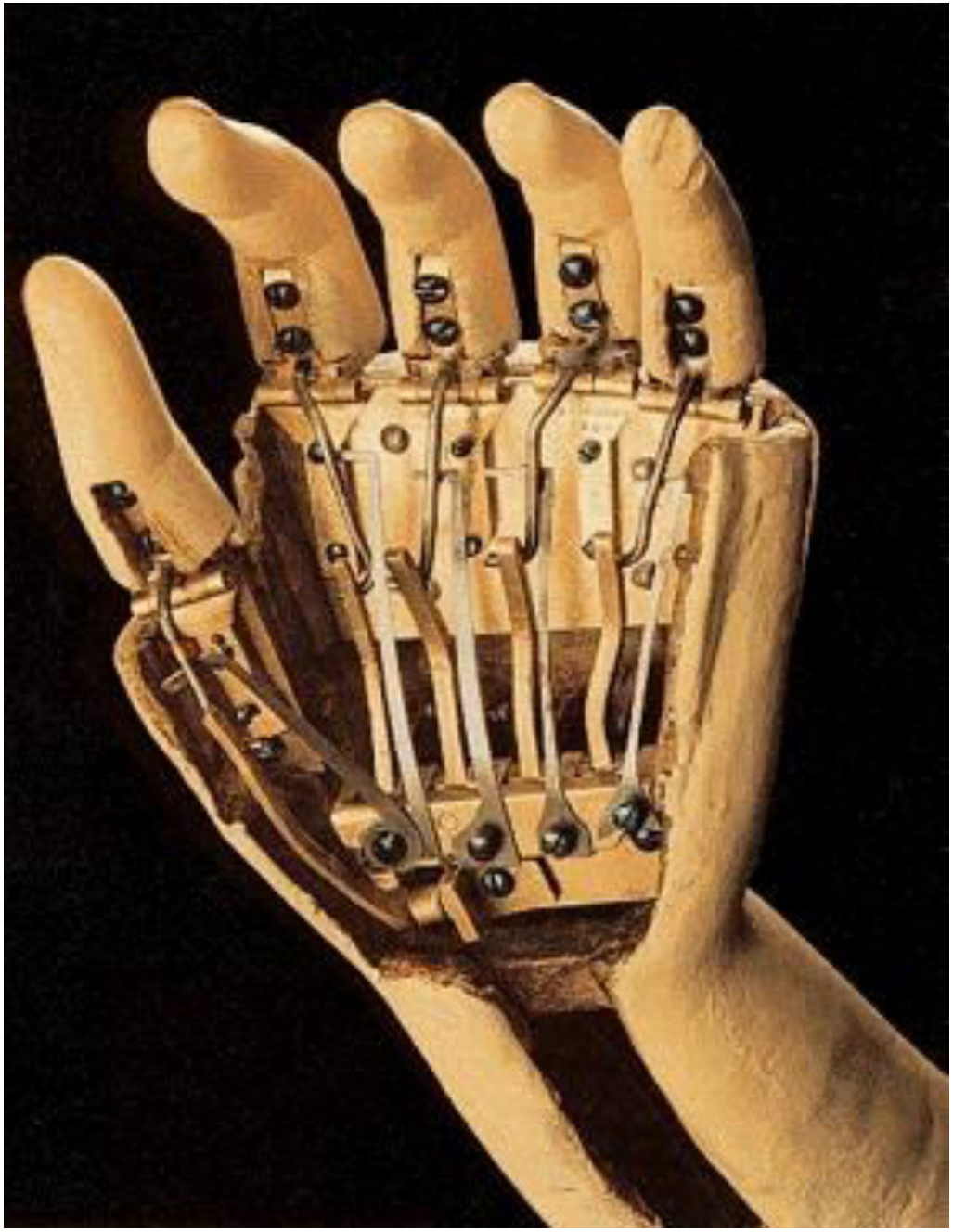
ricostruzione di
M. Mogi Vicentini
(Civico Planetario
di Milano, 2007)



al-Jazarī (circa 1136 – 1206)

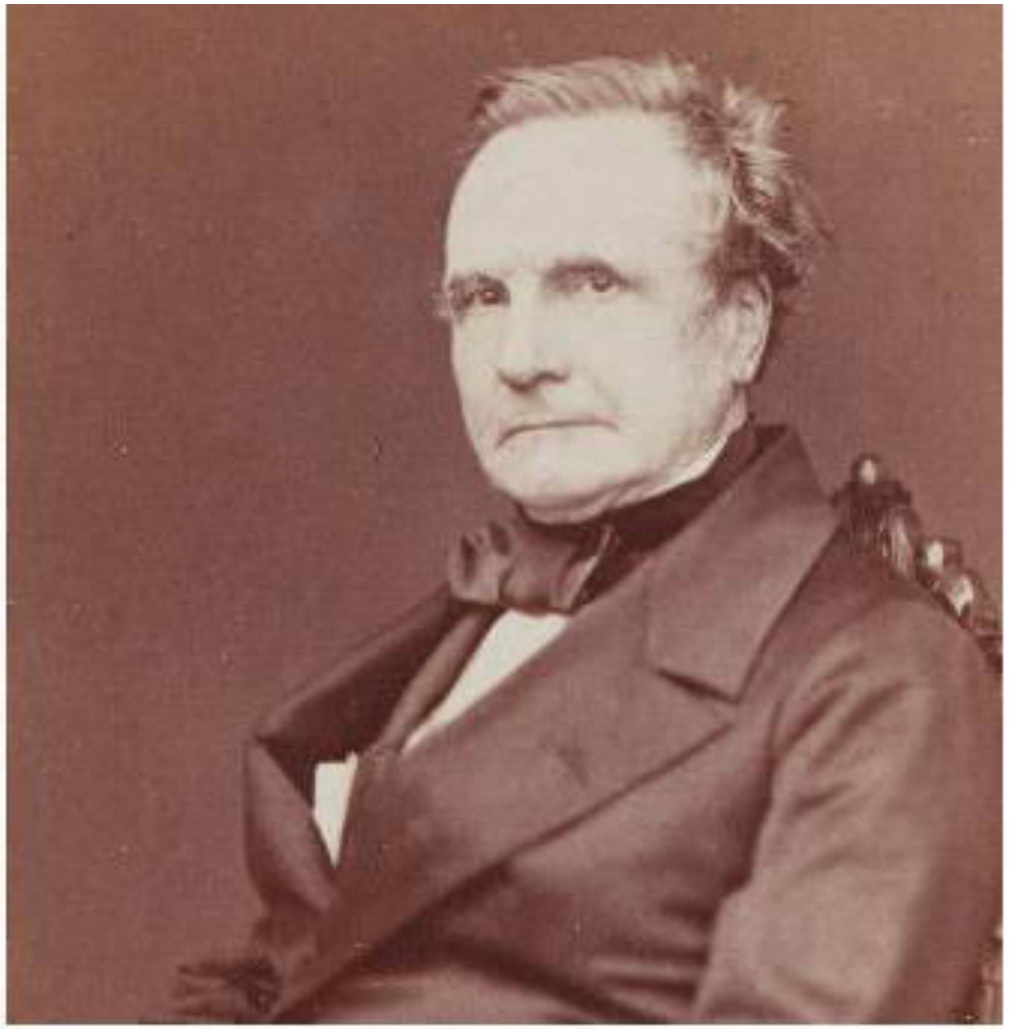
Pierre Jaquet-Droz (1721 – 1790)



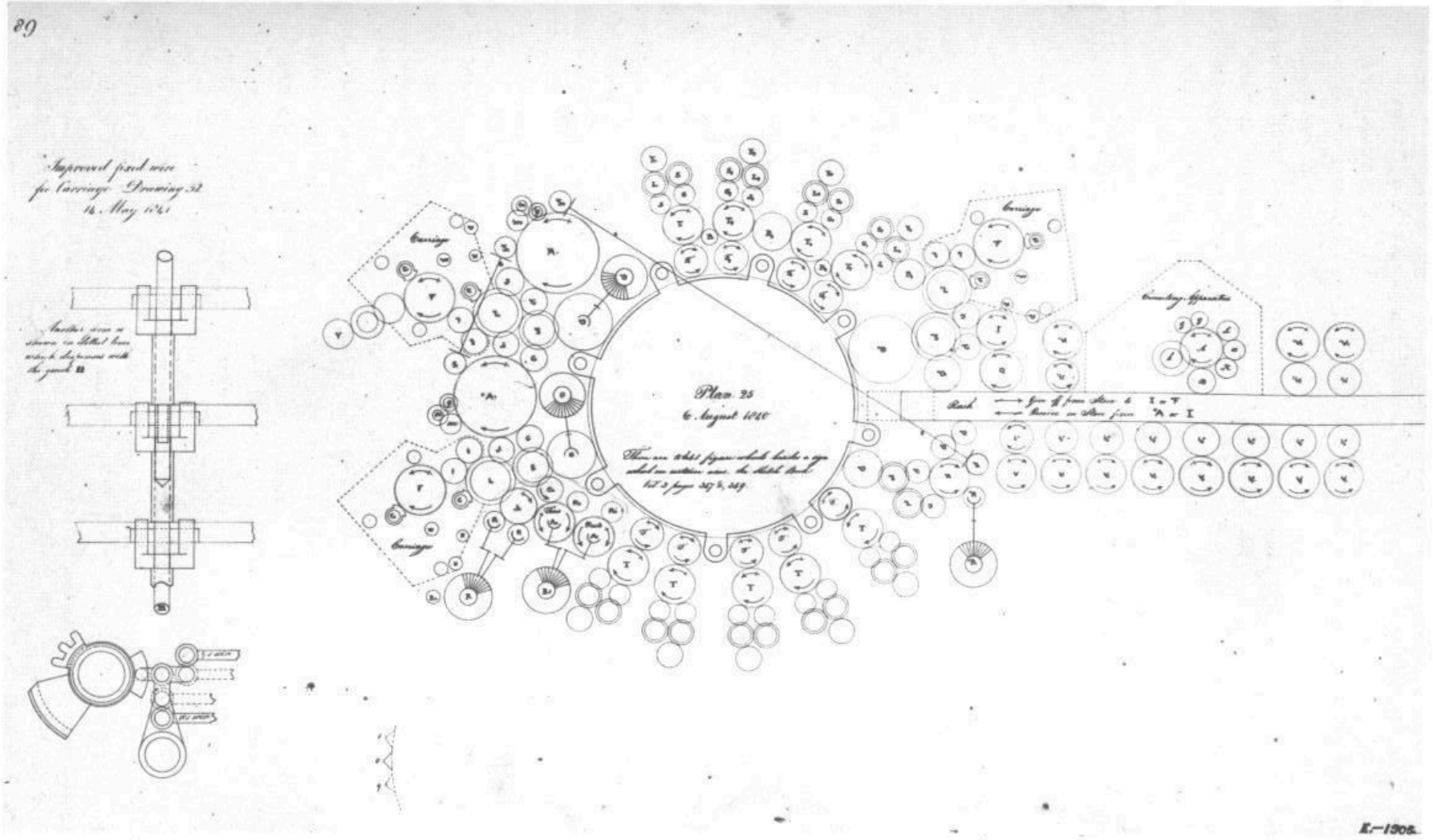


Joseph-Marie Jacquard (1752 – 1834)

Charles Babbage (1791 – 1871)



As soon as an Analytical Engine exists, it will necessarily guide the future course of the science.



The Analytical Engine: piano generale n. 25, 6 agosto 1840

La Macchina Analitica

- Architettura sorprendentemente moderna!
- Separazione logica tra processore e memoria (numerica).
- Unità di calcolo “microprogrammata”.
- Esecuzione a stadi successivi, “in pipeline”!
- Separazione tra istruzioni e dati su distinte unità di lettura.

Così si aggira il problema dell’indirizzamento con indice: la macchina può leggere ripetutamente le operation-card (in un ciclo) mentre legge nuove variable-card sulle quali opererà (sequenzialmente immesse nell’apposita unità di lettura).

Inoltre, le unità di lettura possono funzionare in entrambi i sensi, avanti e indietro, e quindi permettere salti (condizionati).

Quale “linguaggio” fu previsto da Babbage per la sua macchina analitica?

Una sorta di linguaggio macchina, con istruzioni della forma

(calcola) $V_2 \times V_3$
(e immagazzina il risultato nella variabile ricevente) V_4

codificata con Operation-card × e Variable-card (2, 3, 4)

Ada Byron avrebbe poi scritto: $V_4 = V_2 \times V_3$

(come fa per la prima volta nella sua Nota D) ma anche:

$$V_2 \times V_3 = V_4$$

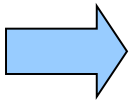
Nel diagramma di Ada per calcolare la successione di Bernoulli:

- non è usato esplicitamente un array, e quindi deve essere preordinata una (lunga e precisa) sequenza di variable-card;
- non sono formalizzate le istruzioni di salto condizionato, ...

ma tutto è descritto precisamente!

- è impostato un calcolo di complessità (numero di operazioni aritmetiche), ma nulla è detto sui progressivi errori numerici.

Punti di forza delle *Note* di Ada:

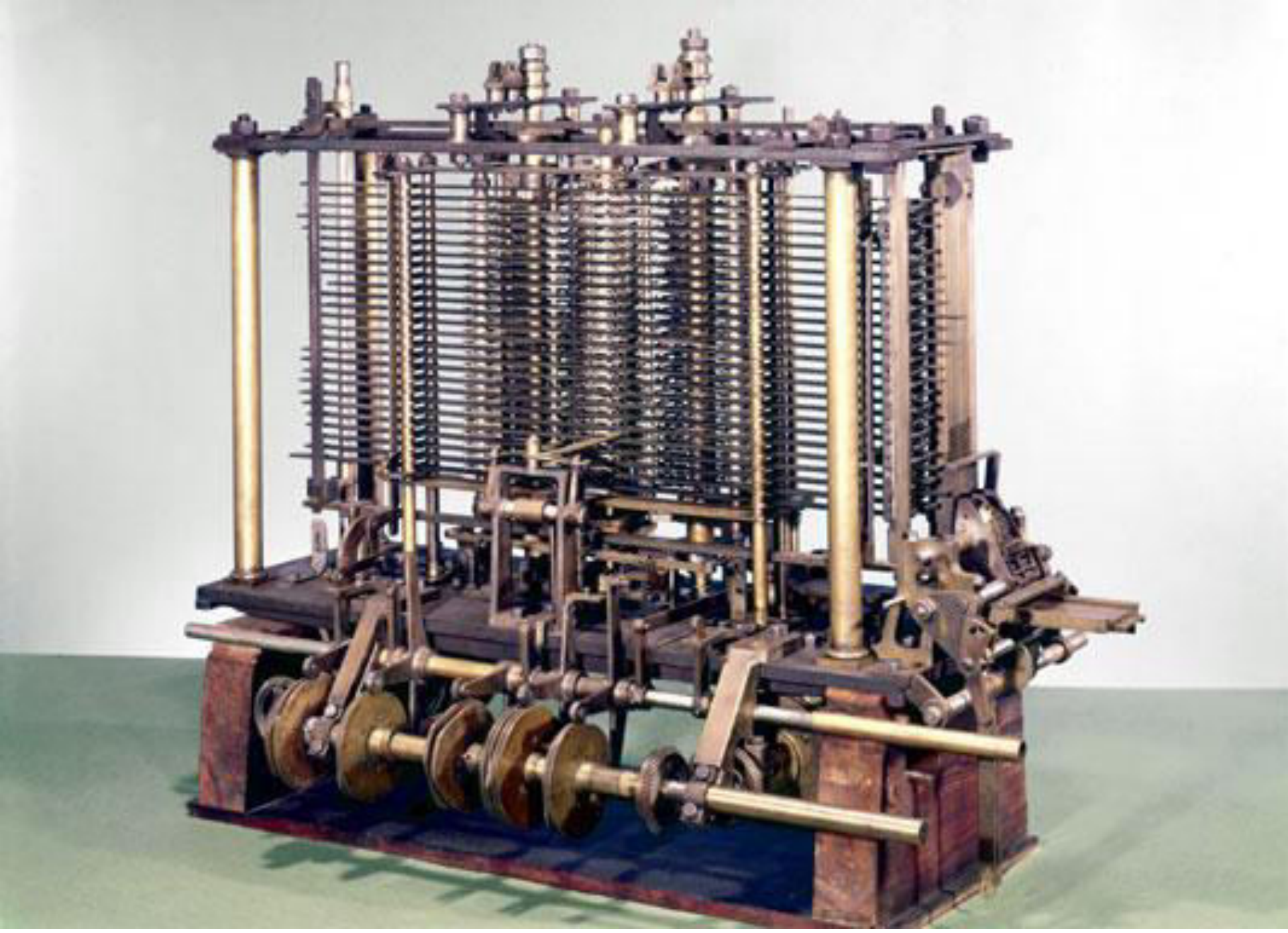
- capacità di sintesi e di penetrazione nelle problematiche epistemologiche;
- la macchina analitica è potenzialmente universale: permette qualsiasi computazione (già intuito da Babbage);
- i numeri possono rappresentare entità che non siano mere quantità o misure  calcolo simbolico !

- Nessuna forma di indirizzamento indiretto.
- Restrizioni imposte da un programma esterno.

Perciò Babbage pensò di includere, tra gli altri sofisticati dispositivi di output, un **perforatore di schede**: non solo per riutilizzare certi risultati numerici, ma anche per consentire alla macchina di produrre i propri programmi!

Così, almeno fino agli anni ’40 del Novecento, la macchina concepita da Babbage fu quella che più si avvicinò a un “**computer universale**” …

Tuttavia, non fu mai costruita!



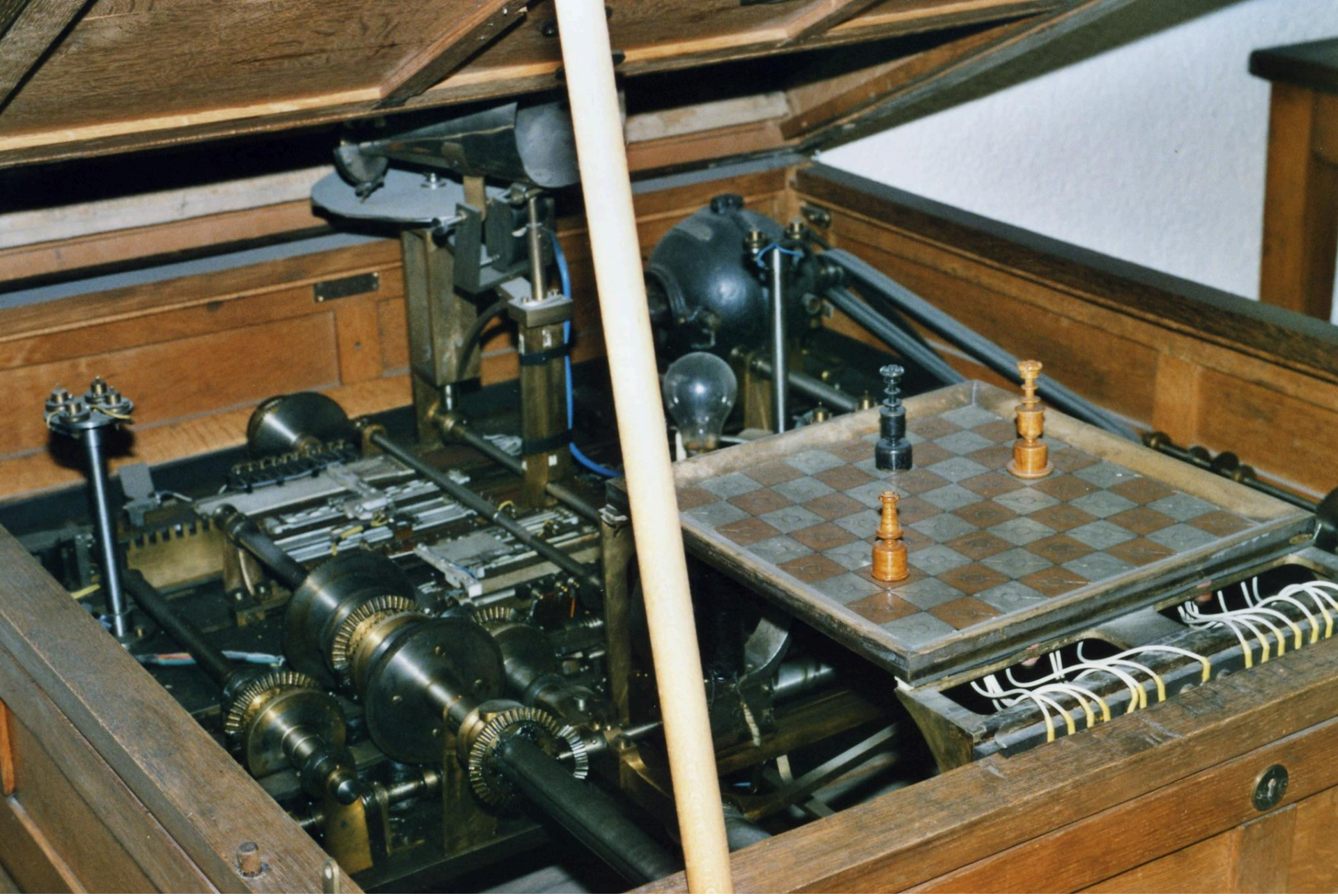
Konrad Zuse

(1910 – 1995)

Le prime macchine costruite da Zuse

Idee che riuscì per primo a realizzare in una macchina:

- meccanizzare l'aritmetica binaria (Leibniz, 1679)
  [studi e tentativi pratici di impiego della numerazione binaria nelle macchine da calcolo: Vartat e Couffignal, 1936; circuiti a relé per realizzare l'algebra booleana: Shannon e Stibitz, 1937]
- controllare i calcoli da programma (Babbage, 1837)
  [unità di calcolo e di controllo separate dalla memoria; meccanismo per accelerare la propagazione dei riporti: lo schema equivale all'addizionatore carry look-ahead]
- usare formati di istruzione con indirizzi di memoria numerici (Ludgate, 1909)
- rappresentare i dati da elaborare in floating-point (Torres y Quevedo, 1914)



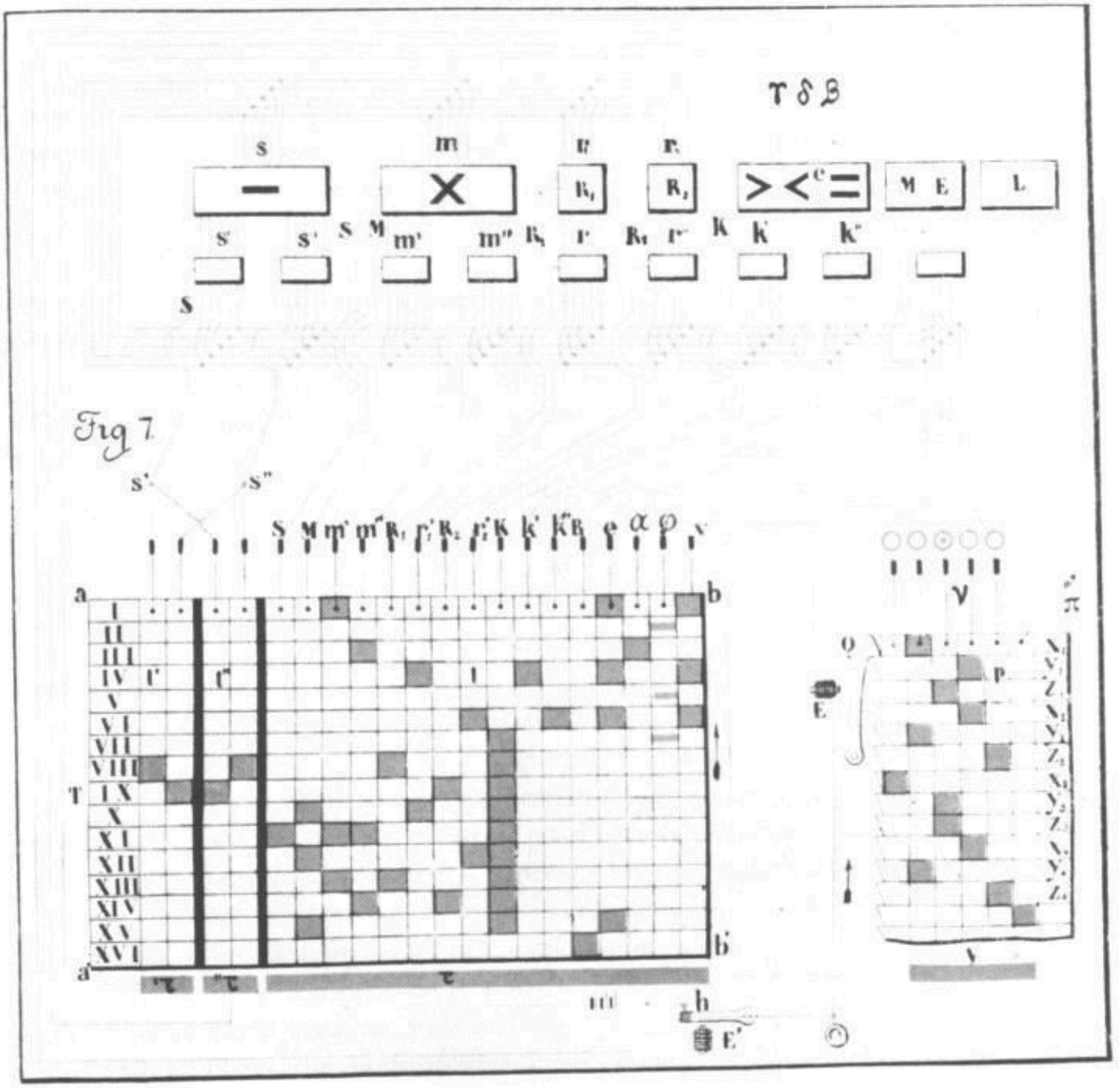
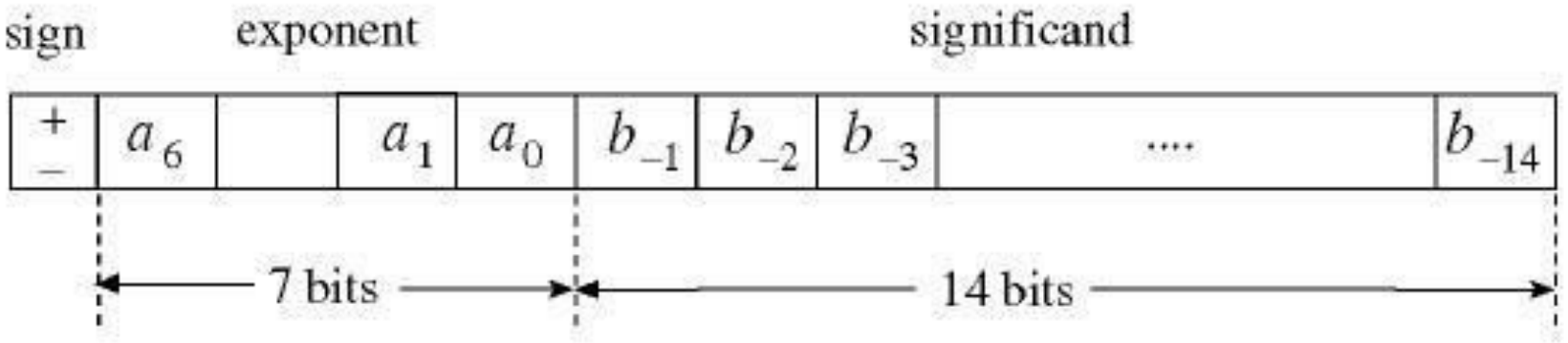


Fig 7.

Z3 (1939-1941)

- Tutto a relé, ma con la stessa architettura dello Z1 (con unità di calcolo e di controllo separate dalla memoria)
- Qualche operazione in più (radice quadrata), introduzione dello hidden bit (in mantissa), trattamento di eccezioni …
- ma soprattutto più veloce e affidabile!
- Costruito a Berlino in Methfesselstraße 7, presentato il 12 maggio 1941, attestato da DVL completo e funzionante il 5 dicembre 1941, distrutto in un bombardamento nel 1944
- fu il primo computer (digitale) controllato da programma (sebbene esterno, a sola lettura) pienamente operativo!
- Zuse pensò già nel 1937 di memorizzare il programma, ma la memoria era costosa, soltanto 64 parole di 22 bit ciascuna, così i programmi erano esterni e perforati a mano su pellicole
- Contiene circa 2600 relé: 600 (stepwise) nell'unità aritmetica, 1800 nella memoria (incluso il multiplexer per la scelta degli indirizzi), 200 nelle altre parti (incluso il dispositivo di lettura)
- Non aveva istruzioni di salto condizionato, né stack (sebbene le operazioni aritmetiche seguissero la notazione postfissa)

Rappresentazione “semi-logaritmica” (floating-point)



Zuse la reinventa e introduce lo hidden bit: è sottinteso **`1.`** davanti alle 14 cifre binarie della mantissa, quindi $b_0 = 1$.
È simile agli attuali standard IEEE.
Occorre una speciale rappresentazione per lo zero …

Esponente (di 2) intero, in complemento a 2: **`−64 (= 0) .. 63 (=`** ∞**`)`**
Overflow: se esponente del risultato ≥ **`63`** allora risultato = ∞
Underflow: se esponente del risultato ≤ **`−64`** allora risultato = **`0`**
Eccezioni: $\infty - \infty$, **`0`** × ∞, ∞ × **`0`**, **`0`** / **`0`**, ∞ / ∞ risultato indeterminato
Un hardware apposito le rileva, accende una lampadina e ferma …

Con tale dispositivo e un numero sufficiente di parole di memoria indirizzabili, un *loop* di operazioni aritmetiche può simulare una qualsiasi MdT (su nastro limitato)!

Il calcolo della radice quadrata nello Z3

Idea naïf: con tutti i bit a 0, il risultato è approssimato (per difetto) partendo dal bit più significativo: si prova con 1, e se il quadrato supera il radicando allora si rimette a 0 …

Come si può fare in modo efficiente, evitando il quadrato? C'è una variante binaria dell'algoritmo di "completamento del quadrato" (Rafael Bombelli, bolognese, metà del '500) che, senza perdere generalità, assume il radicando *a*:

$$1 \leq a < 100_2$$

Qualsiasi numero positivo può infatti essere scritto come

$$a \cdot 2^p$$

con p pari, e quindi basterà moltiplicare il risultato per $2^{p/2}$

L'algoritmo

- r ← 1 (corrisponde al bit nascosto nello Z3)
- e ← $a - 1$ (n = numero di bit della parte frazionaria
- per k = 1, 2, …, n della mantissa; è 14 nello Z3)
  - b ← $2 \cdot e - (2 \cdot r + 2^{-k})$
  - se b ≥ 0, allora (il k-esimo bit della mantissa è messo a 1)
    - r ← $r + 2^{-k}$
    - e ← b
  - altrimenti (il k-esimo bit della mantissa è lasciato a 0 …)
    - e ← $2 \cdot e$ (… e comunque $2^{-k} \cdot e$ non aumenta!)

- Ogni volta che e è assegnato, se vale 0 l'algoritmo può terminare
- Alla fine, il risultato (per difetto) è r , mentre e è l'errore scalato:

$$r^2 + 2^{-n} \cdot e = a$$

Un esempio con $n = 8$

- Calcolare la radice di $1.10110001 \cdot 2^5 = 11.0110001 \cdot 2^4$
- $a = 11.0110001$, $r = 1.00000000$, $e = a - 1 = 10.0110001$

k	$b = 2 \cdot e - (2 \cdot r + 2^{-k})$ [in blu il nuovo valore di e]	r
1	$100.110001 - 10.1 = 10.010001$ (>0)	1.10000000
2	$100.10001 - 11.01 = 1.01001$ (>0)	1.11000000
3	$10.1001 - 11.101$ (<0)	1.11000000
4	$101.001 - 11.1001 = 1.1001$ (>0)	1.11010000
5	$11.001 - 11.10101$ (<0)	1.11010000
6	$110.01 - 11.101001 = 10.100111$ (>0)	1.11010100
7	$101.00111 - 11.1010101 = 1.1000111$ (>0)	1.11010110
8	$11.000111 - 11.10101101$ (<0)	1.11010110

- Il risultato (per difetto) è $r \cdot 2^2$, esatto sarebbe $2^2 \cdot \sqrt{(r^2 + 2^{-8} \cdot e)}$

Un altro algoritmo (Newton-Raphson)

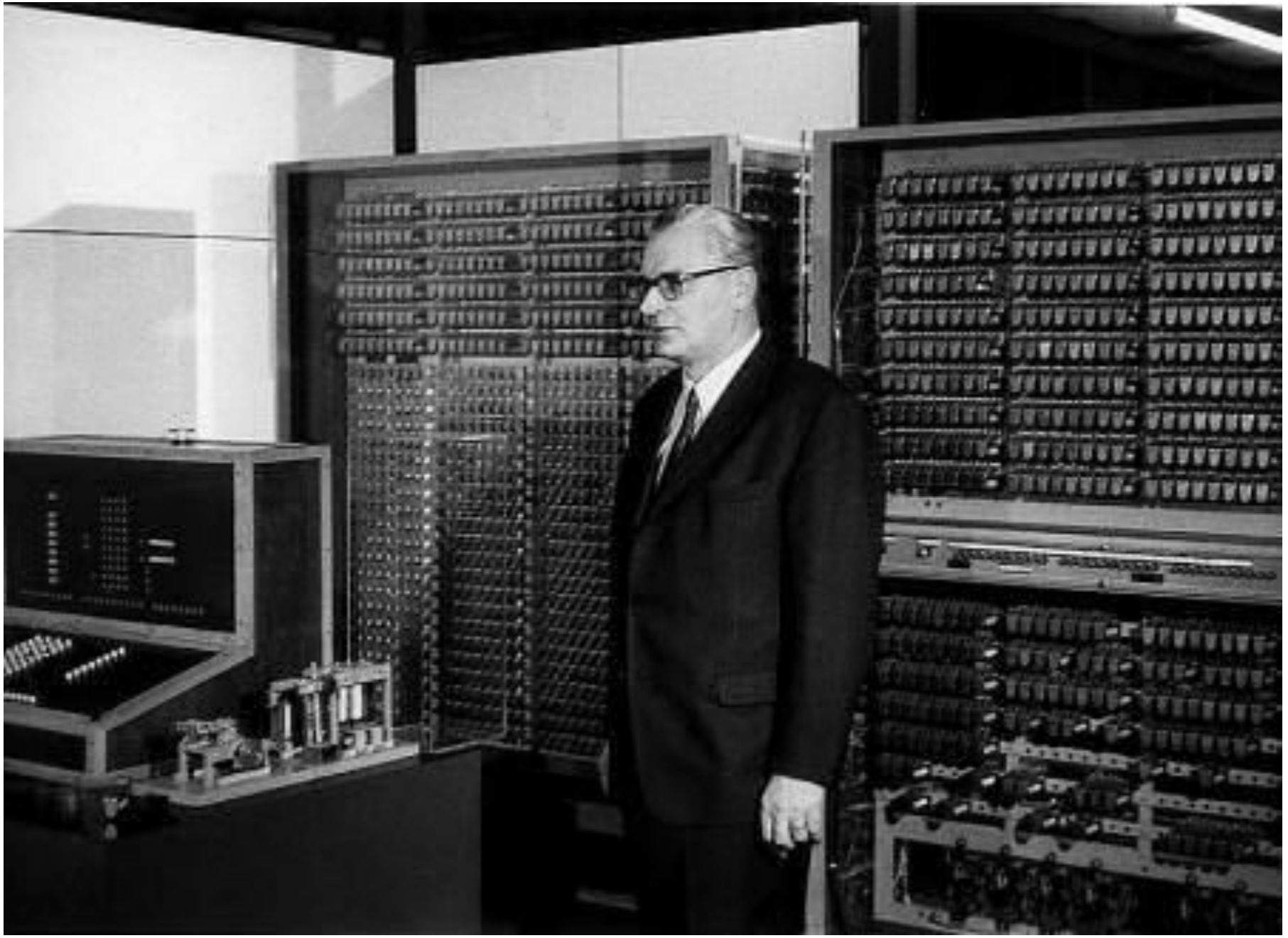
Formula ricorrente già impiegata da Erone di Alessandria (probabilmente ripresa dalla civiltà babilonese):

$$x_0 = a / 2 \qquad \text{(basta positivo, ad es. 3/2)}$$
$$x_{n+1} = (x_n + a / x_n) / 2$$

Si ferma quando scende sotto una tolleranza max relativa.
Nel caso dell'esempio:

$$x_1 = 1.11011000$$
$$x_2 = 1.11010110$$

Convergenza quadratica, anziché lineare: ad ogni passo, il numero di cifre corrette tende a raddoppiare.
Ma con l'algoritmo di Zuse le operazioni sono più semplici e realizzabili in modo efficiente!



Nel 1943/44, in quella che avrebbe dovuto essere la sua tesi di dottorato (sulla costruzione di circuiti per il calcolo), Zuse aveva sviluppato una notazione a tre indirizzi, come quella di Babbage

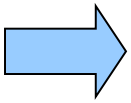
⇒ Starre Pläne (programmi rigidi, lineari)

ma concludeva avvertendo la necessità di uno schema formale (di programmazione) = modello + linguaggio:

- sia per formulare algoritmi numerici
- sia, soprattutto, per trattare problemi di natura combinatoria (logici) di più alto livello (scacchi)

⇒ Unstarre Rechenpläne

“Concentrai i miei sforzi soprattutto sui problemi logici che stanno sotto agli usuali calcoli numerici …”



Plankalkül = sistema formale per la pianificazione dell’elaborazione

1945: “Lo scopo del Plankalkül è fornire una descrizione puramente formale di qualsiasi processo computazionale.”

Il Plankalkül comprende in sé tante idee, ispirate dal calcolo proposizionale (Aussagenkalkül) e dal calcolo dei predicati (Prädikatenkalkül) di Hilbert, ma non soltanto …

Però la definizione originale del Plankalkül presenta delle inconsistenze e pure parecchie ambiguità, che devono essere risolte se si vuole implementare il linguaggio!

Prima di mettere da parte questo lavoro per dedicarsi ancora al miglioramento dello Z4 e alla progettazione di nuove macchine, Zuse completò un corposo manoscritto contenente procedure assai più complesse di qualsiasi altra mai scritta prima. Tra le tante cose vi erano algoritmi:

- di ordinamento
- per decidere la connettività di un grafo rappresentato da una lista di archi (coppie di nodi)
- per l'aritmetica intera in notazione binaria
- per l'aritmetica floating-point (radice quadrata inclusa)
- per decidere se una data proposizione logica è ben formata (sintatticamente) e se ha parentesi ridondanti
- per gli scacchi (anni prima di Shannon e Turing!)
  - la funzione di valutazione considera il solo materiale
  - non è trattata correttamente la presa "en passant" …

Chi inventò il computer?

- Il computer fu una conquista collettiva, che abbracciò due continenti, l'Europa e l'America, e almeno una dozzina d'anni, dal 1936 al 1948 …

- Giugno 1948: Manchester "Baby" fu il primo calcolatore elettronico a programma memorizzato internamente (la memoria era a CRT di Williams).

Non aveva possibilità di indirizzamento indiretto, ma di salti condizionati e programmi automodificanti.

Turing lo programmò nel settembre di quell'anno …