Stonehenge

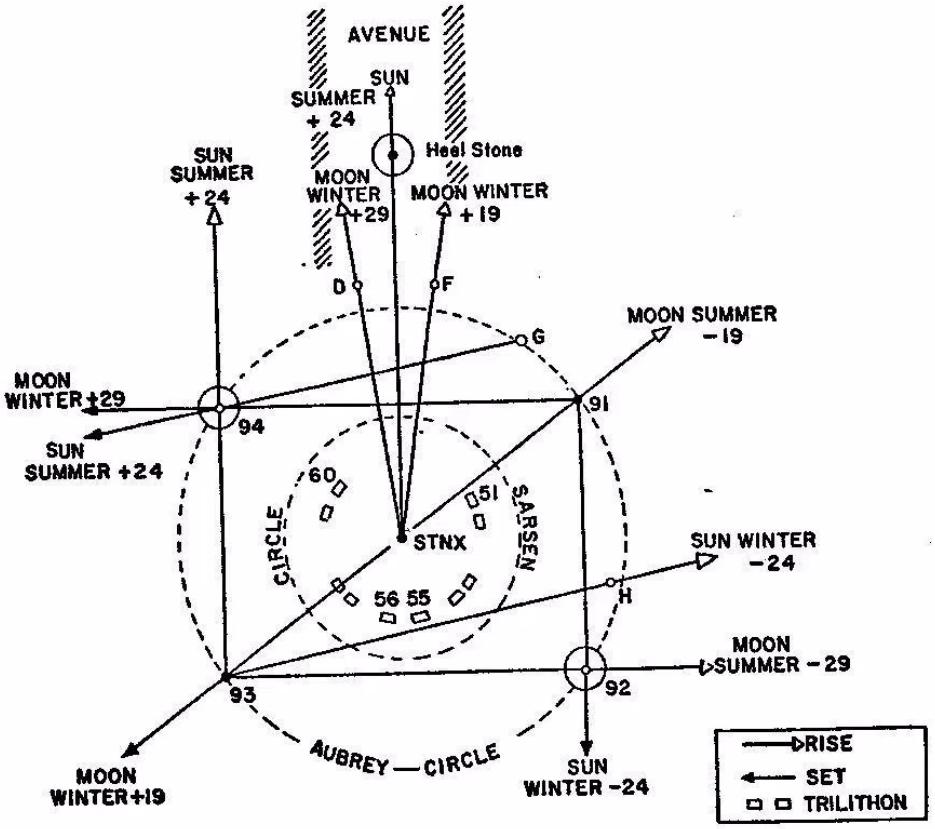
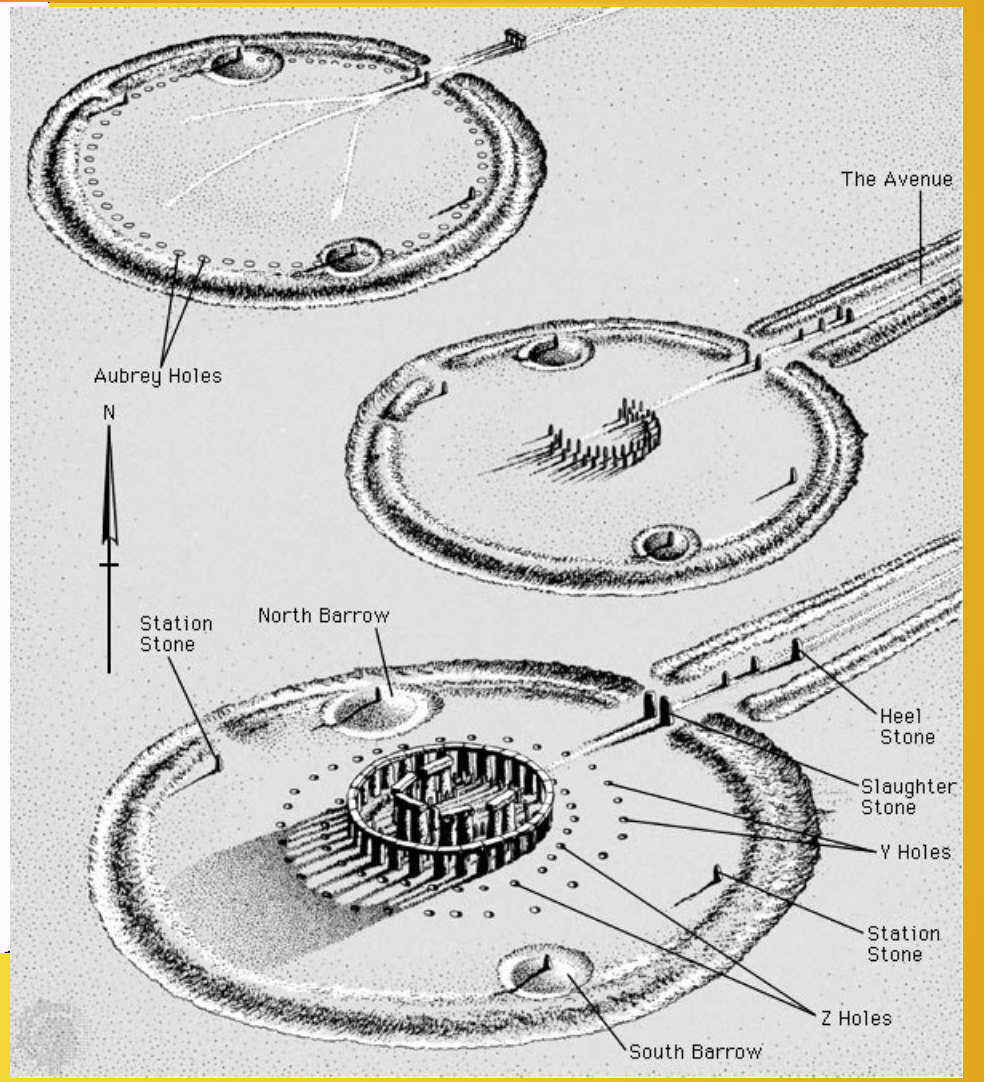
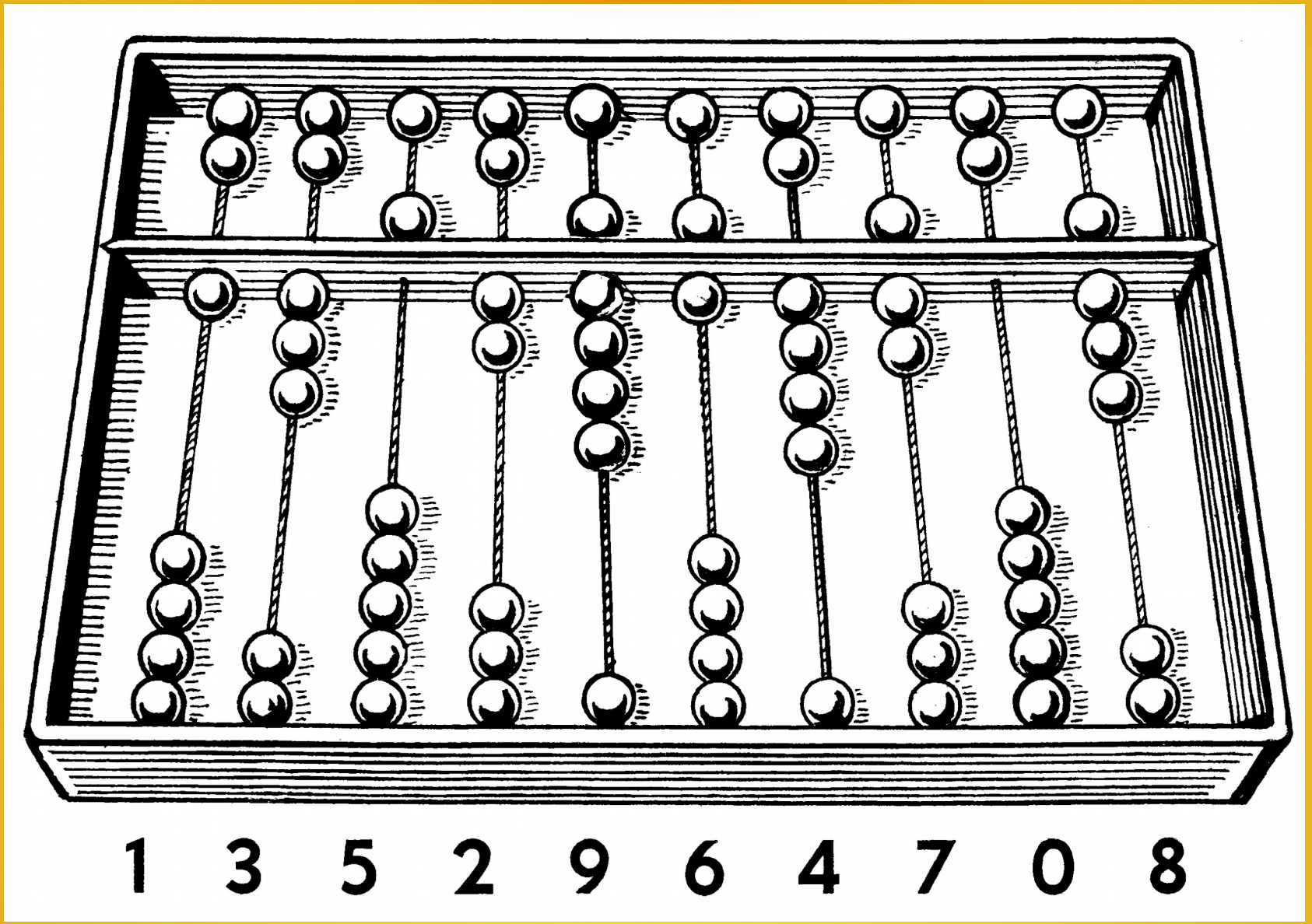


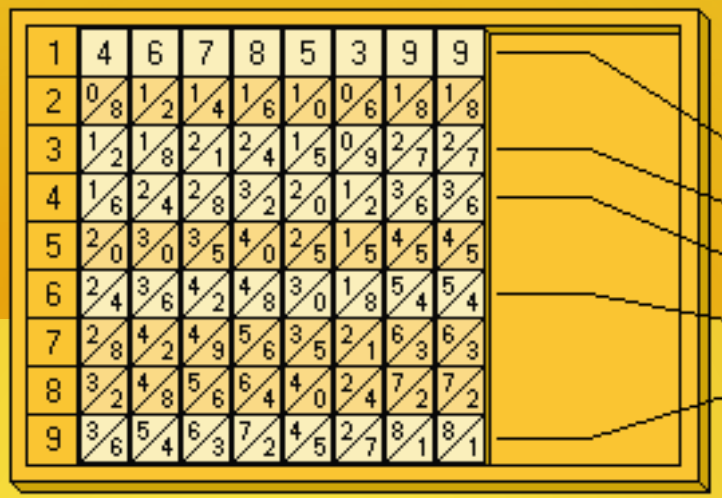
Fig. 1. Schematic plan of Stonehenge



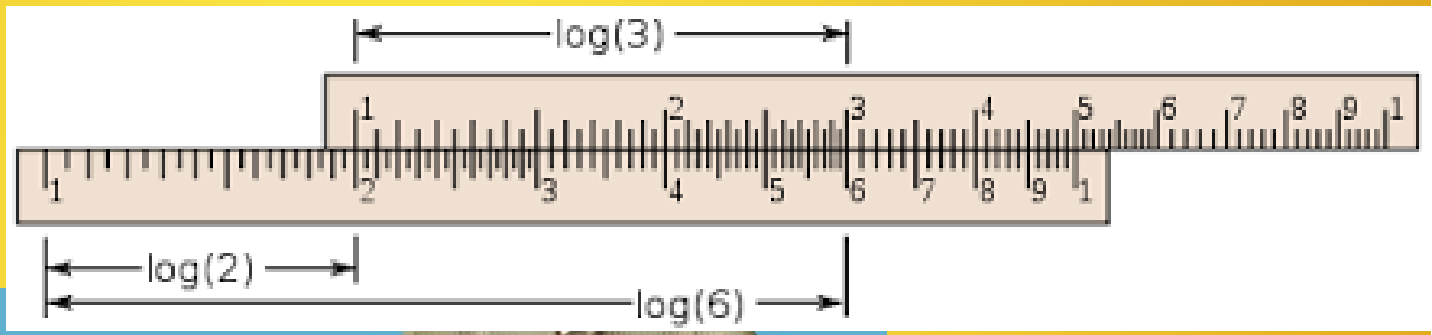
ABACUS

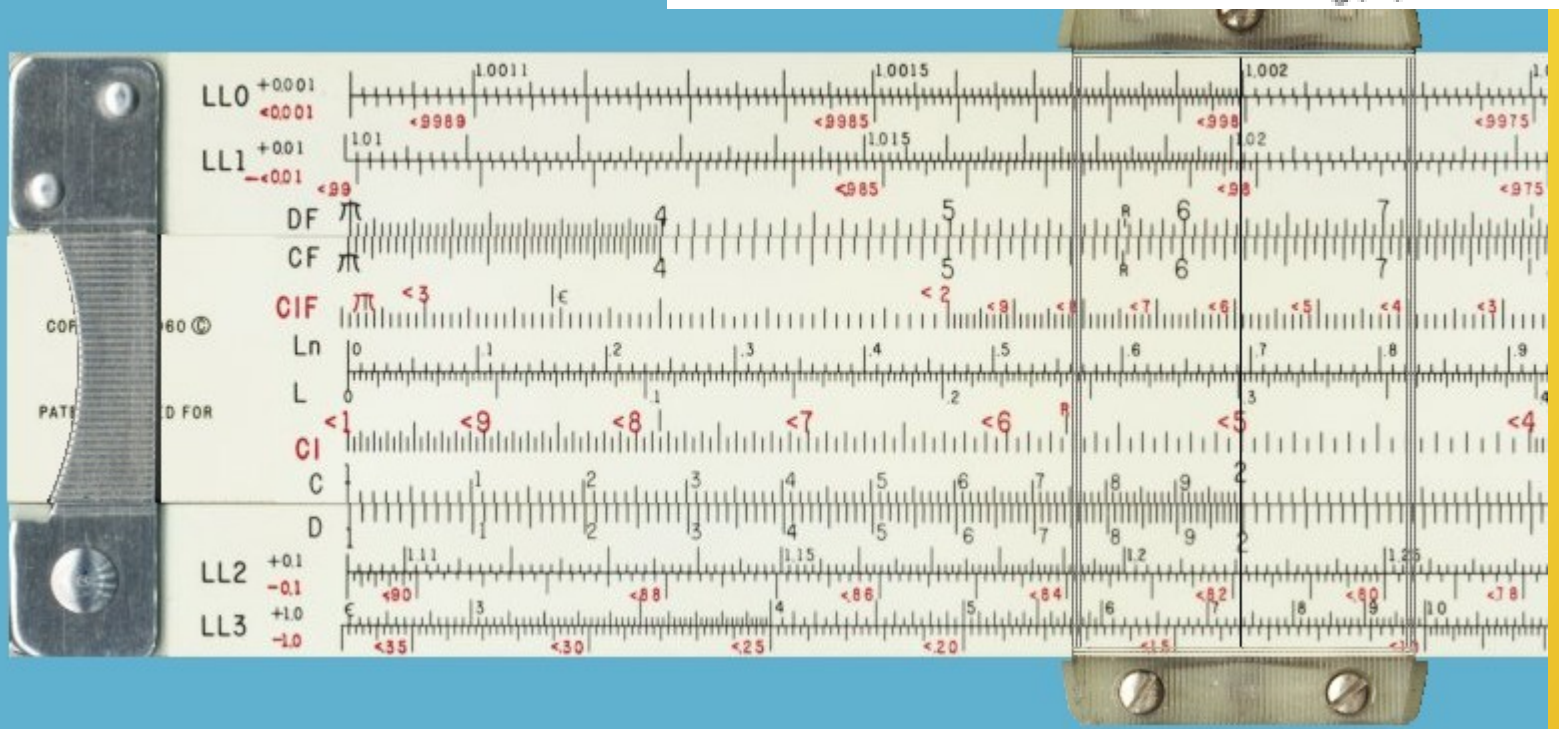


John Napier



46785399
×96431
46785399
140356197
187141596
280712394
+421068591
4511562810969





Blaise PASCAL

Blaise Pascal (1623-1662)

Francia filozófus és matematikus.
A tízes számrendszer jegyeit egy körtárcsa kerületére írva megoldja az átvitel (carry) automatizálását s ezzel a számlálás, valamint az erre alapuló összeadás illetve kivonás gépesítését.

Fig. 1.

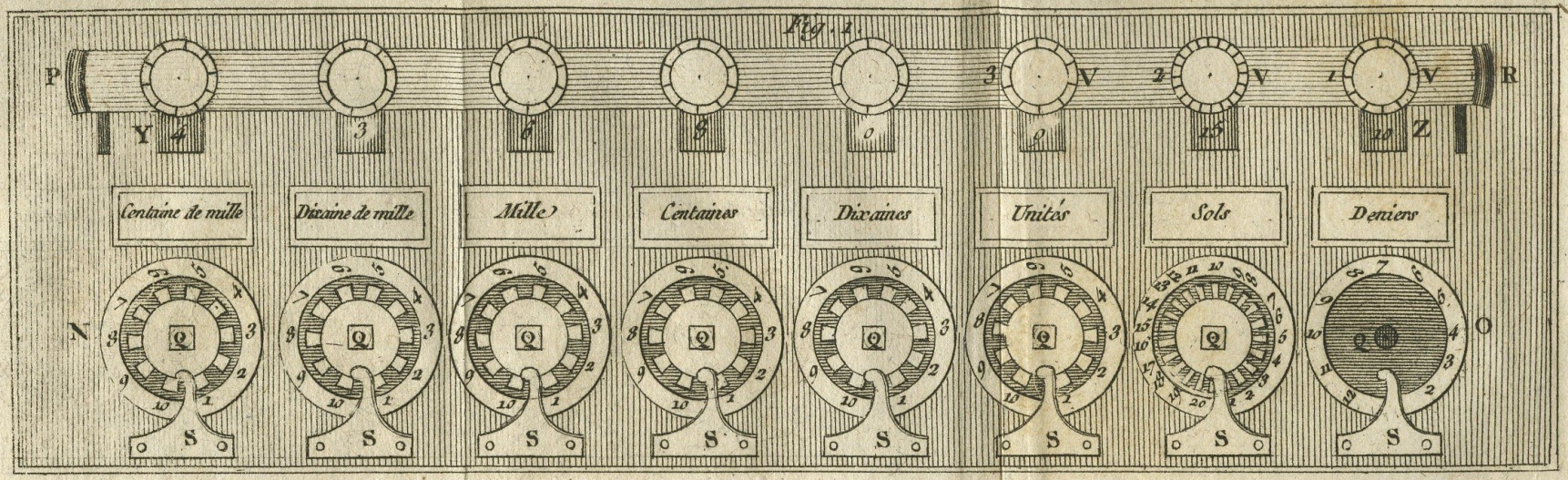
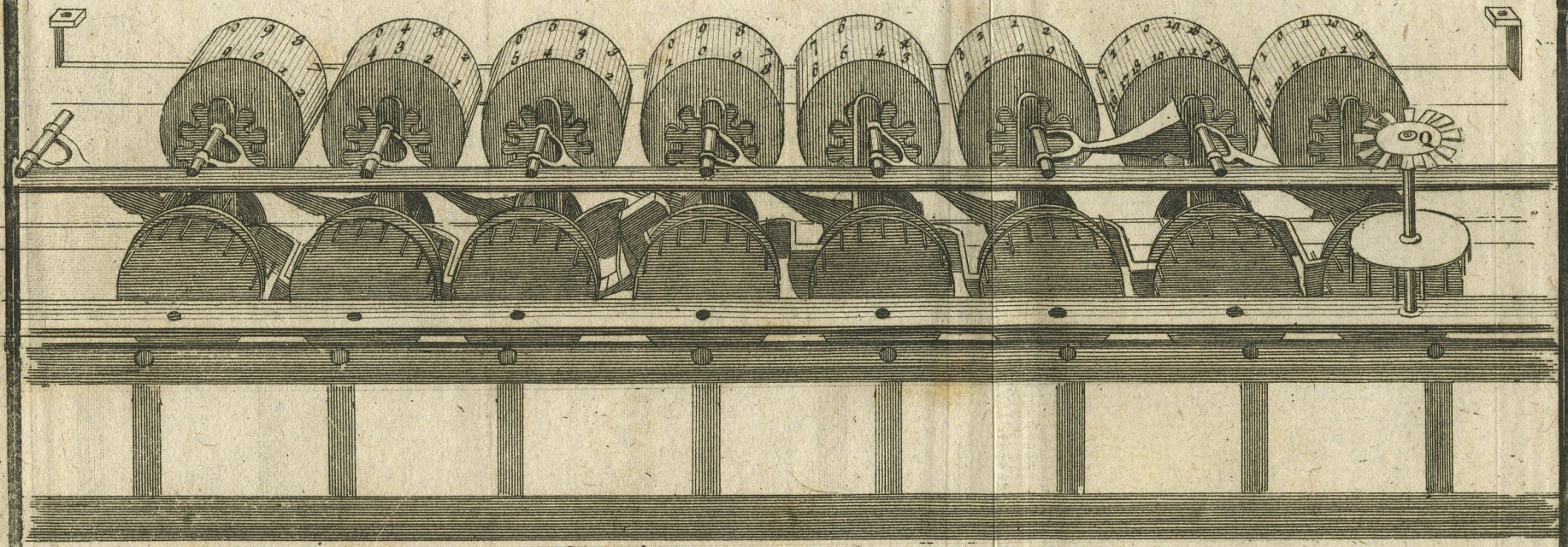


Fig. 2.



P. B. Bradel del et Sculp.

2.

Fig. 3.

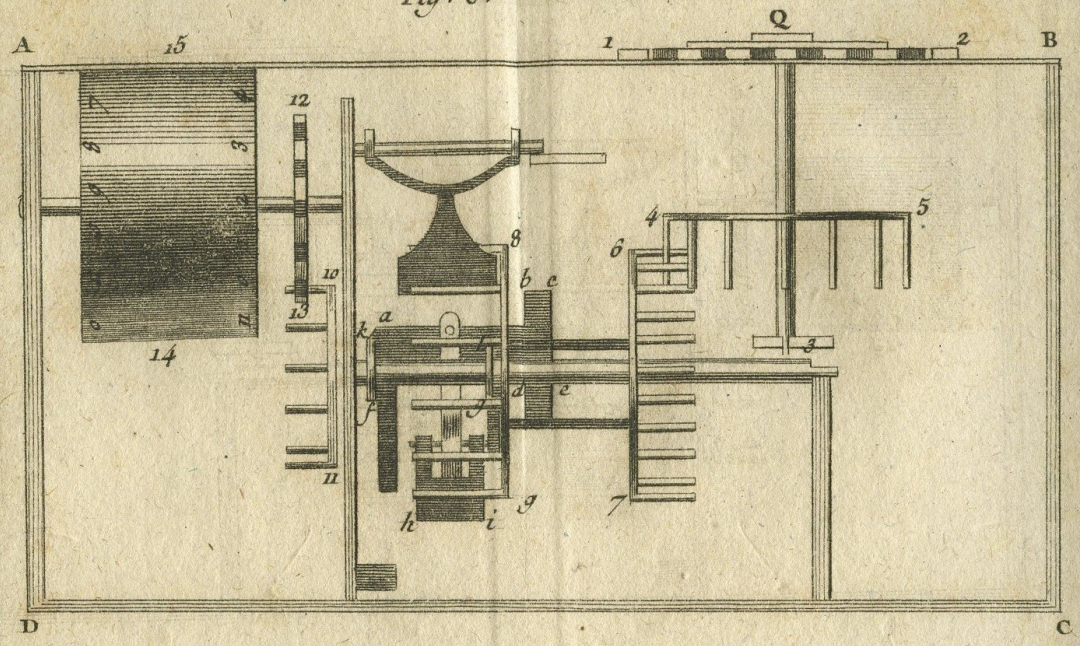


Fig. 6.

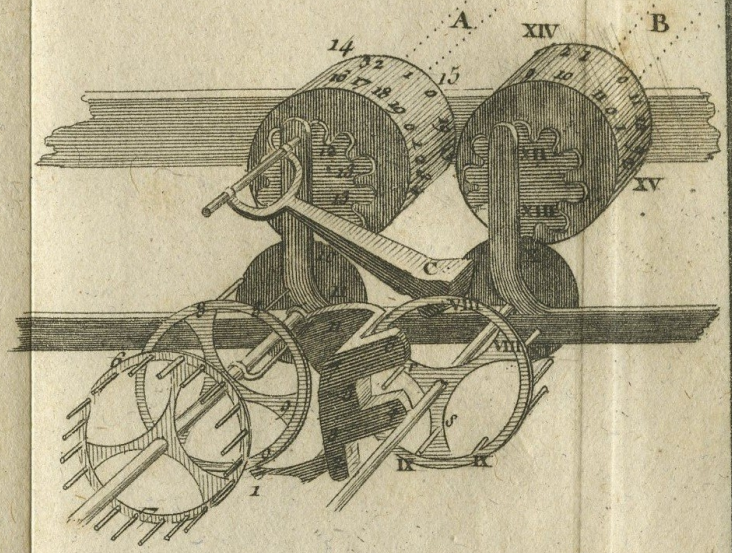


Fig. 4.

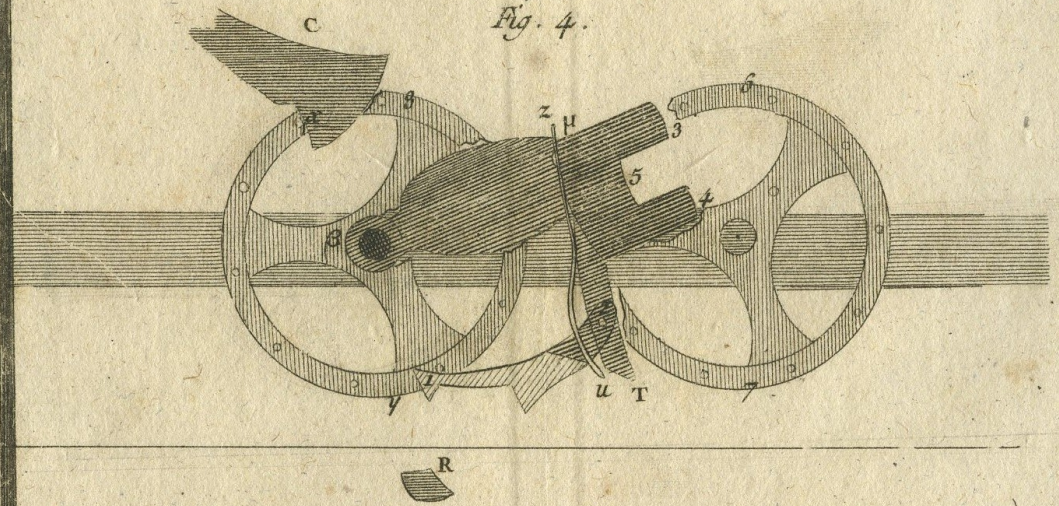
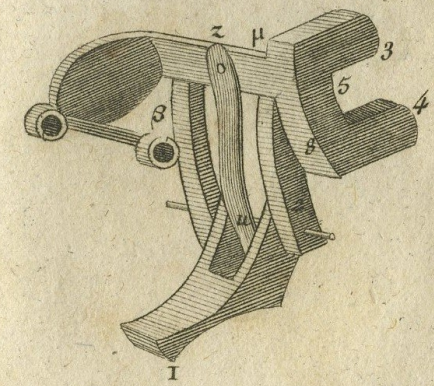
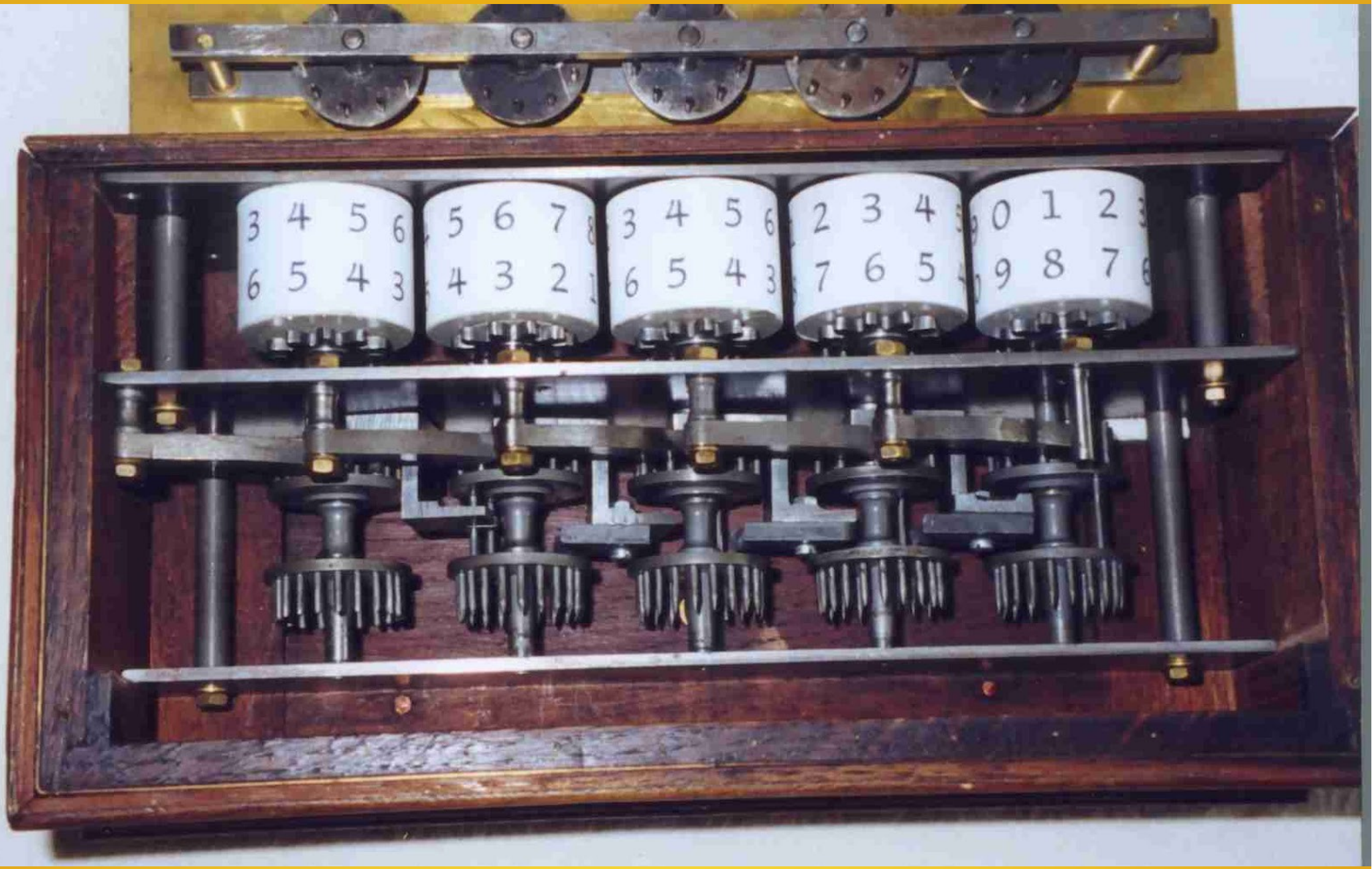
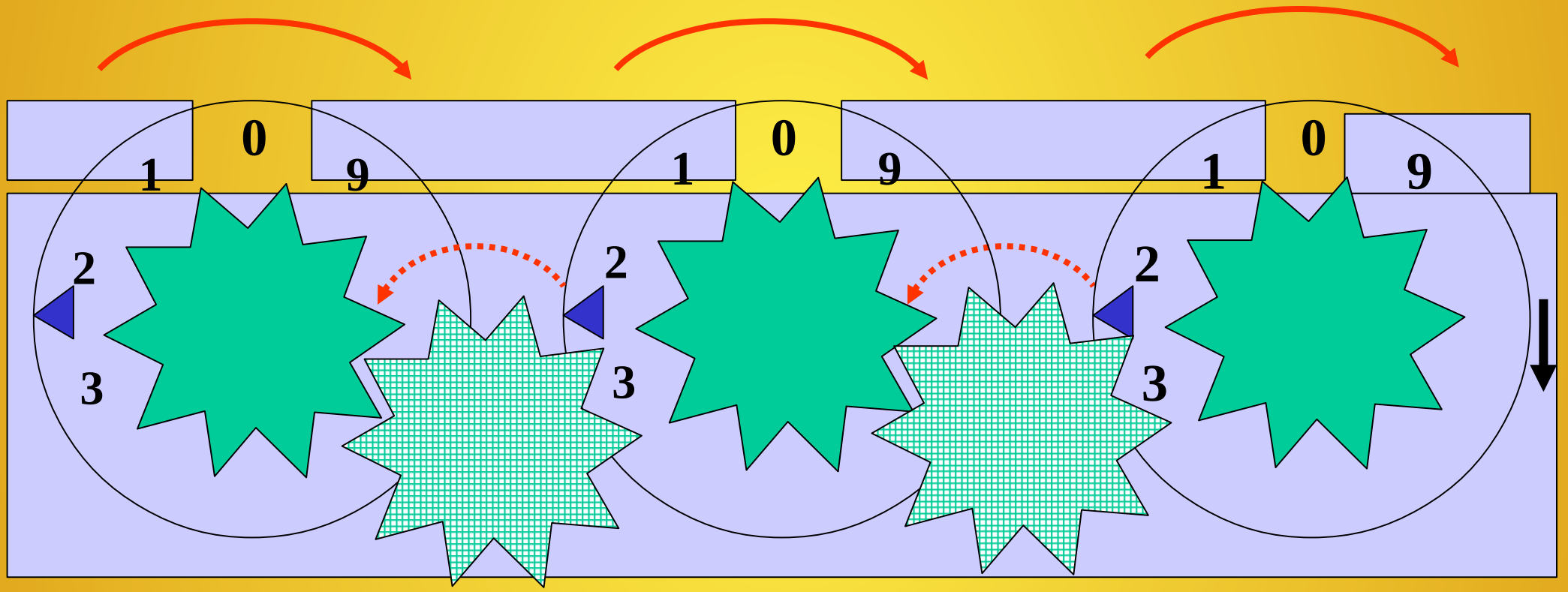


Fig. 5.




3 4 5 6
6 5 4 3
5 6 7 8
4 3 2 1
3 4 5 6
6 5 4 3
2 3 4 5
7 6 5 4
0 1 2 3
9 8 7 6

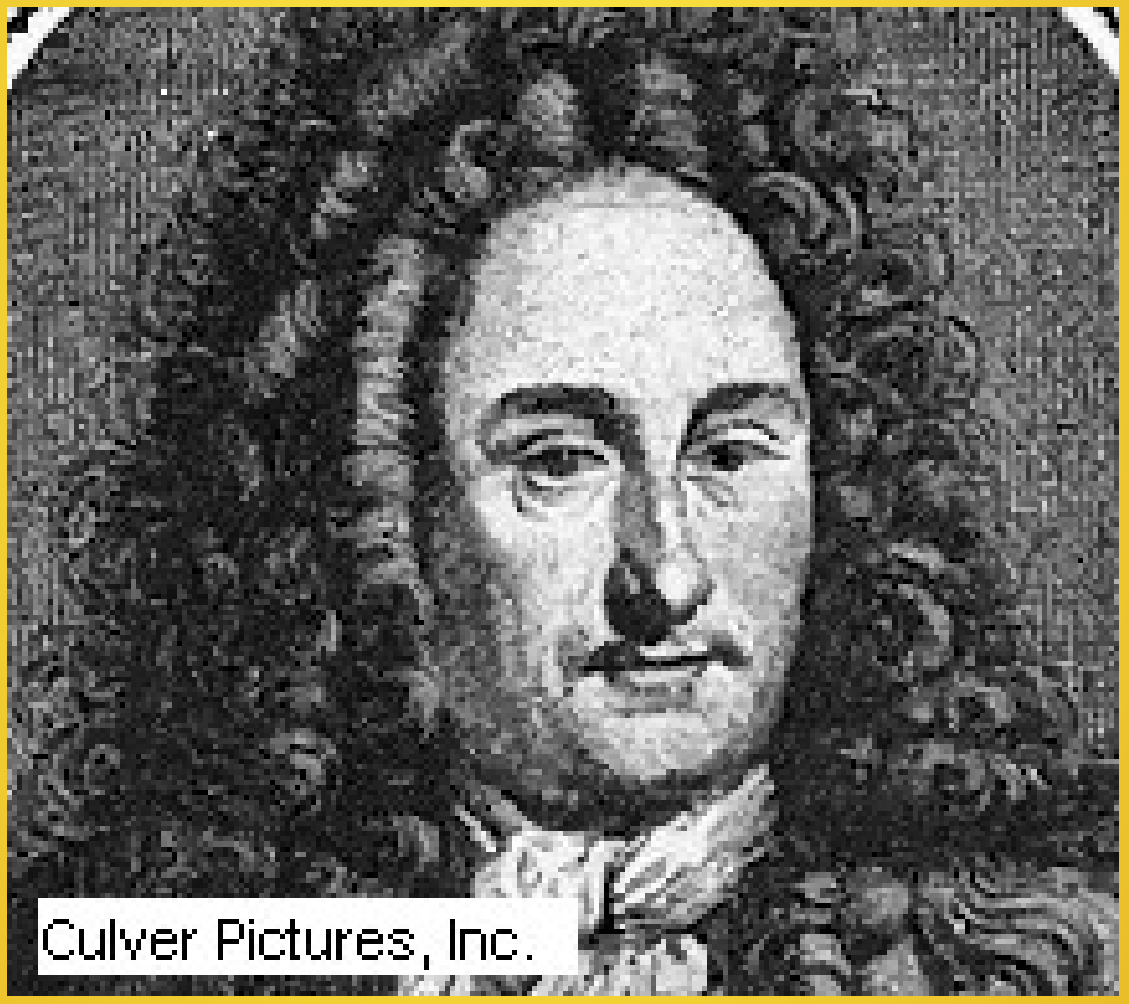
PASCAL KALKULÁTORA (1642)



Az „átvitel-jegy” automatikusan tovább vonul !

A kallantyúval számlálásra is alkalmazható

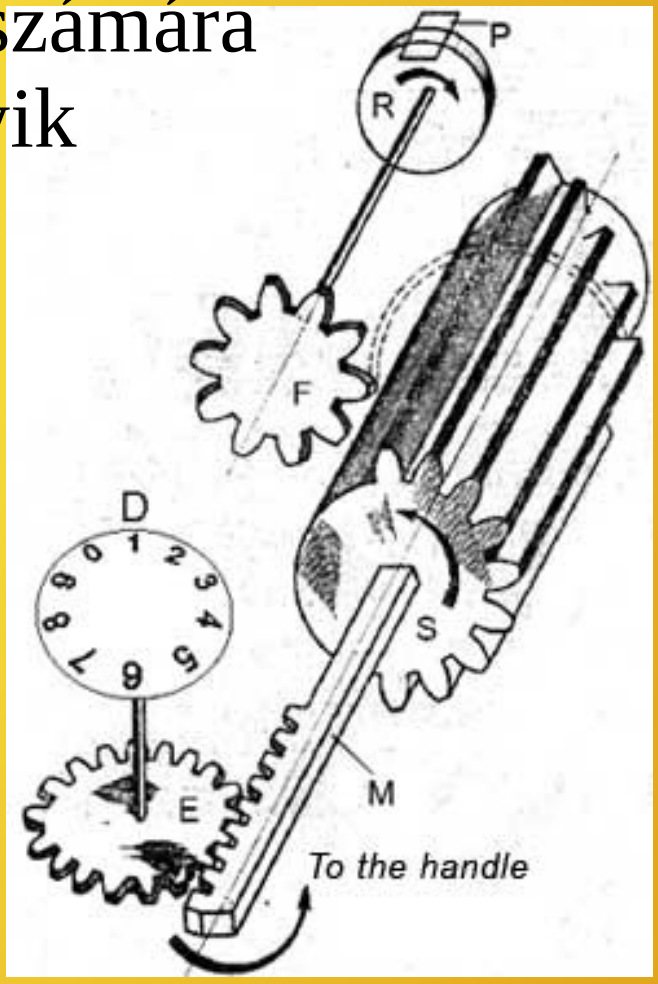
Gottfried Wilhelm LEIBNIZ



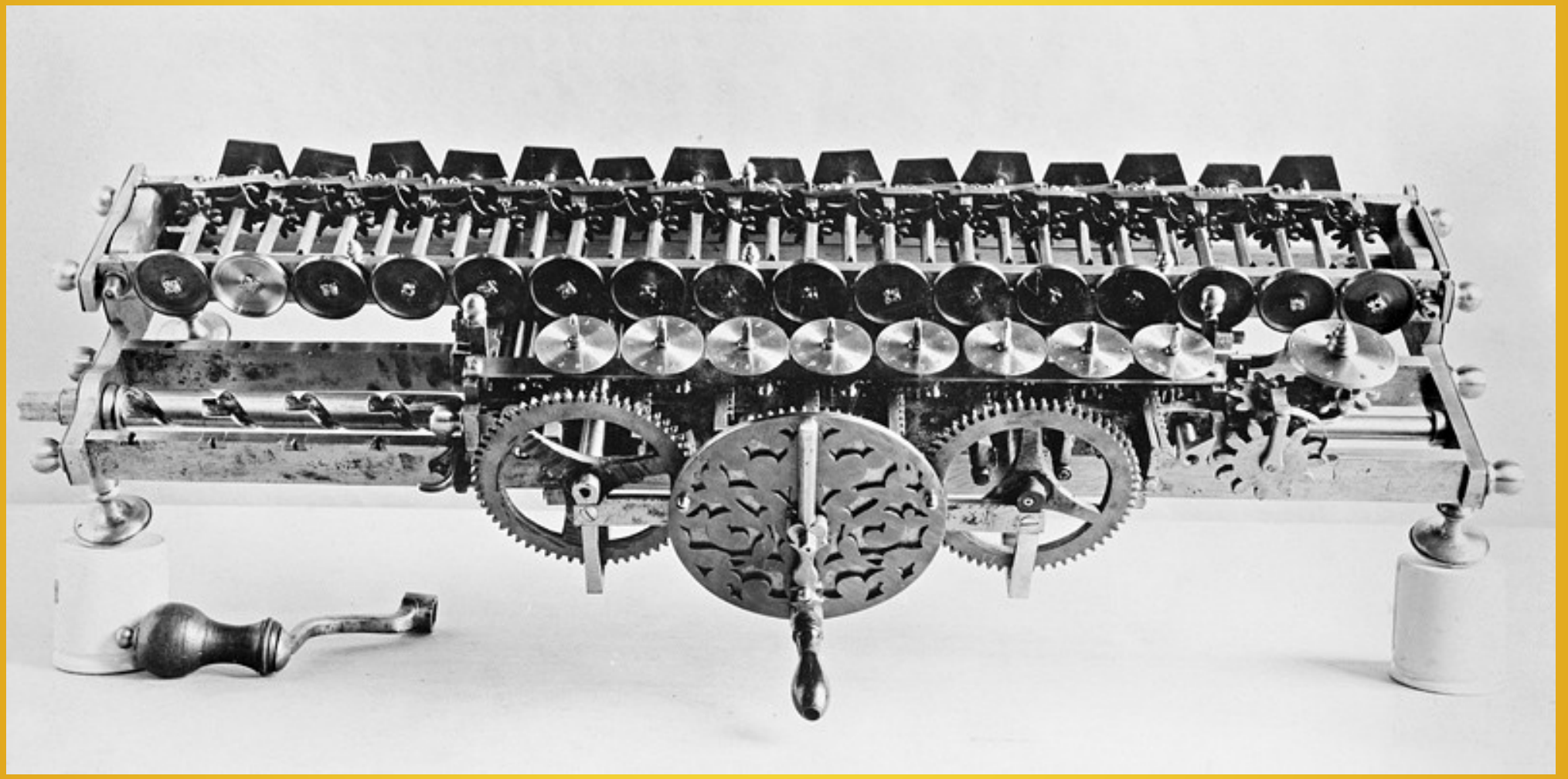
Gottried Wilhelm von LEIBNI(T)Z 1646-1716

Német filozófus és matematikus.

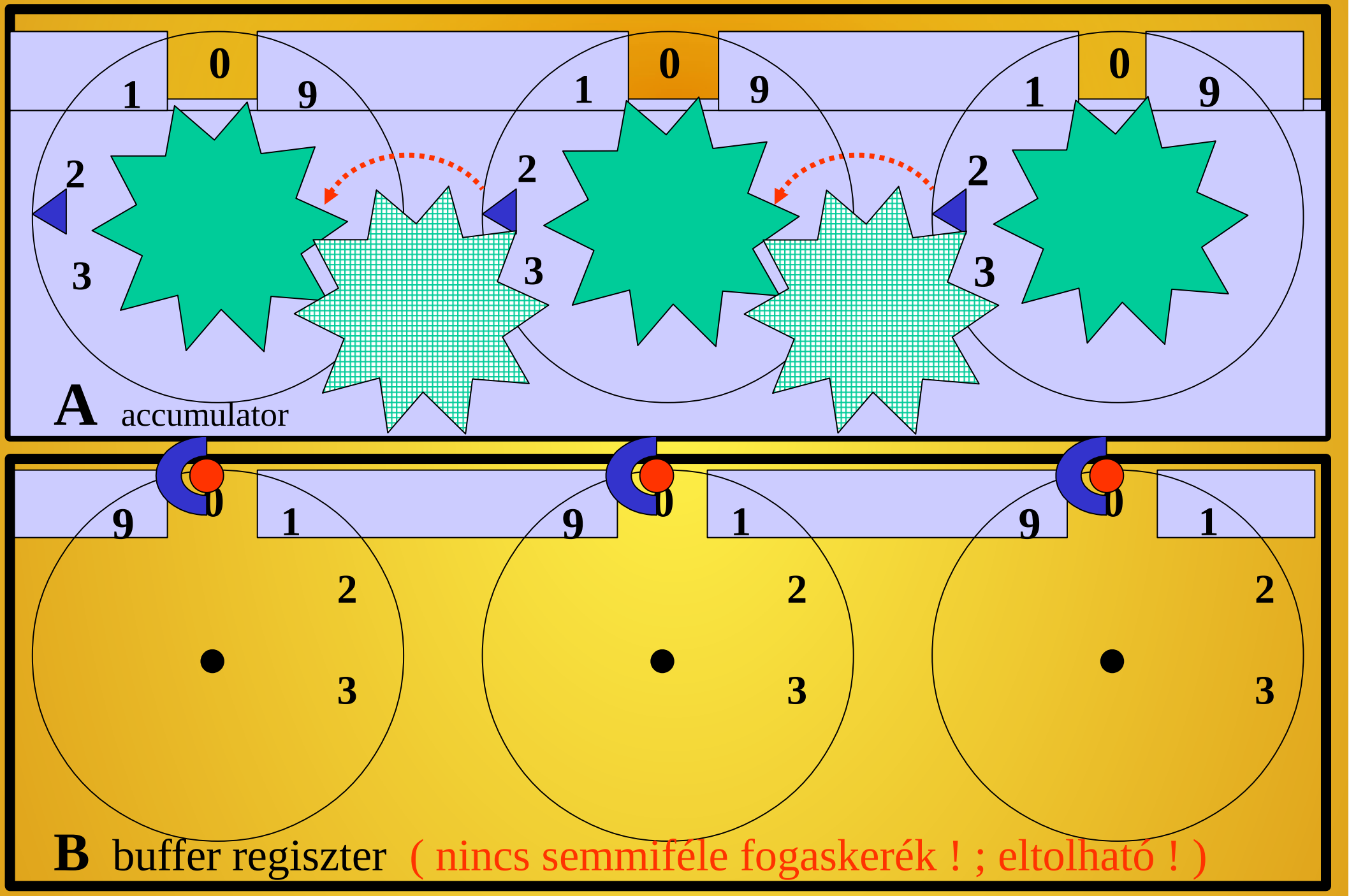
Pascal kalkulátorát továbbfejleszti, bevezetve az alapműveletek mindkét operandusa számára egy-egy regisztert, melyek közül az egyik eltolható. Így megvalósíthatja a tízzel, majd ismételt összeadás ill. kivonás segítségével a tetszőleges számmal való szorzást ill. osztást is.



LEIBNITZ KALKULÁTORA (1692)



LEIBNITZ KALKULÁTORA (1692)



AZ ANALÍZIS MEGJELENÉSE

Leibnitz és Newton kialakítják az infinitezimális számítást.

Kialakul az analízis. Fel lehet írni differenciálegyenleteket.

Még a legegyszerübbeknek sincs zárt alakú megoldása:

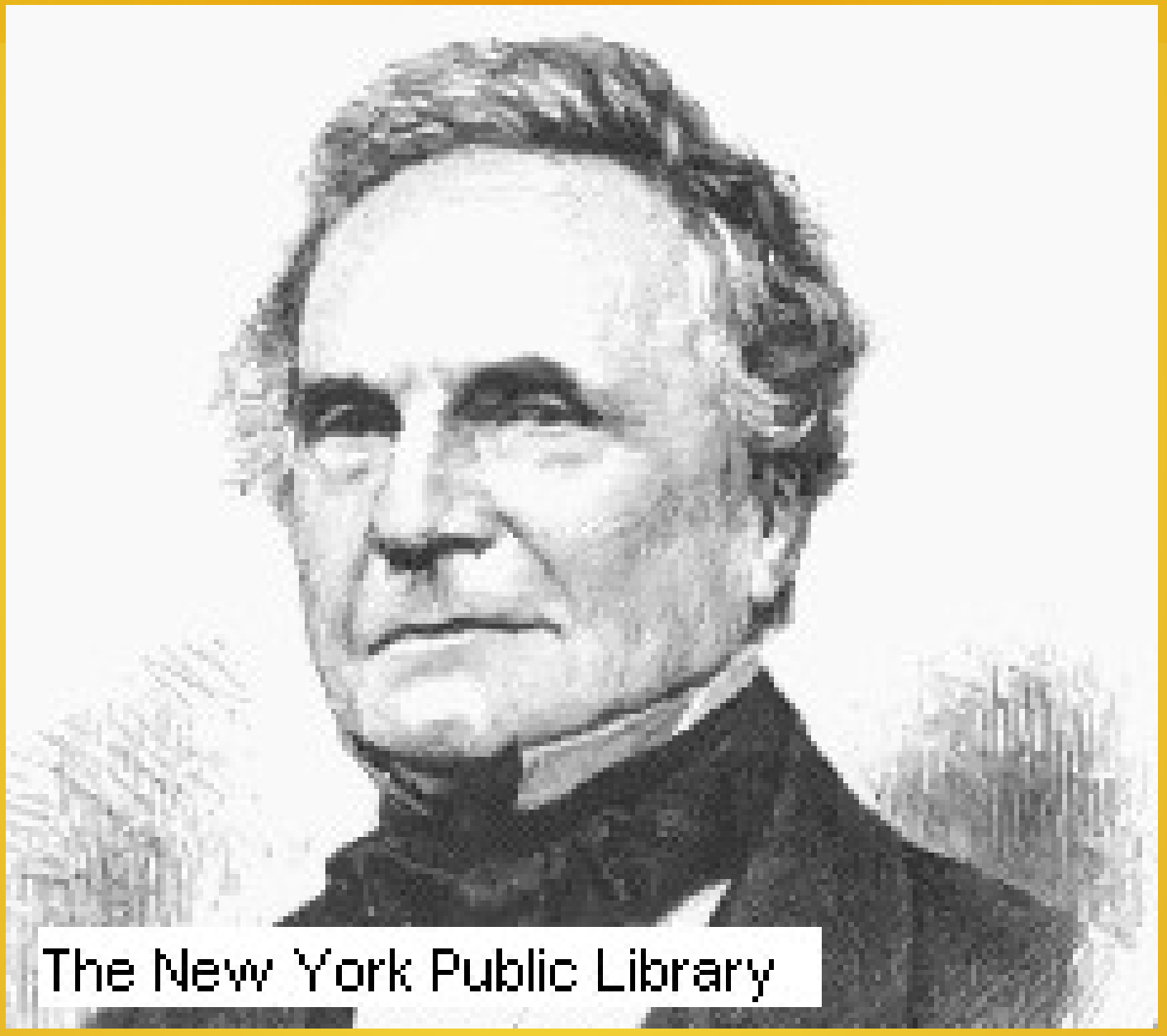
y’ + y = 0 megoldása $y = e^{-x}$; y nem számítható ki x-ből
y” + y = 0 „ y = sin x ; cos x „

Ezeknek a függvényeknek az értékét táblázatok adják meg.

Ezeket a táblázatokat meg kellett csinálni. (Logar- tábla)

Babbage ezt a táblázatkészítést akarta automatizálni.

Charles BABBAGE



Charles Babbage 1792- 1871

A polinómok táblázatolására kifejleszti és megépíti az u.n. Differencia Gépet.

Ennek továbbfejlesztése képen az egymáshoz kapcsolt hat összeadó helyett egyetlen

kalkulátort (aritmetikai egységet) és sok tároló rekeszt tartalmazó memóriát javasol, melyből az adatok lyukkártyákon tárolt utasítások nyomán jutnak az aritmetikai egységbe ill. abból vissza a memóriába. Ez a mai számítógép őse.

TAYLOR SOR

Babbage tudta, hogy
minden folytonos függvény
hatványsorba fejthető:

f(x) = f(0) + f’(0).x/1! + f’’(0).x^2/2! + …..

Pl.:

sin (x) = x - x^3/3! + x^5/5! - x^7/7! + …..

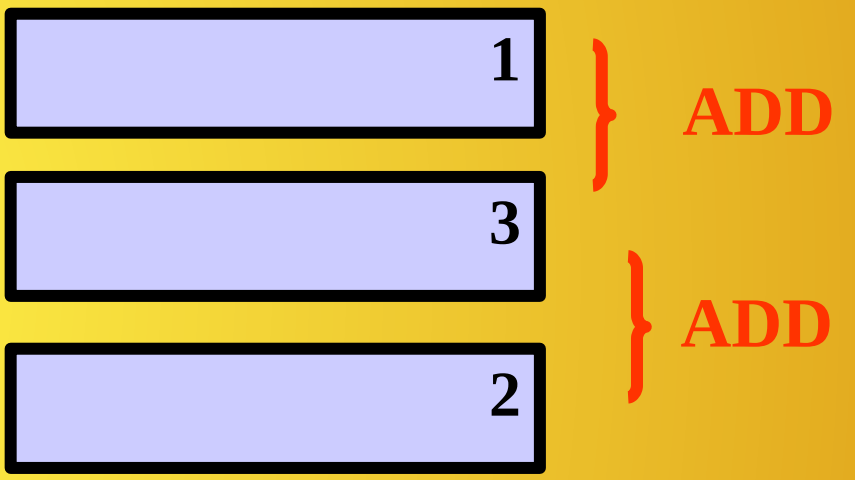
DIFFERENCIA GÉP

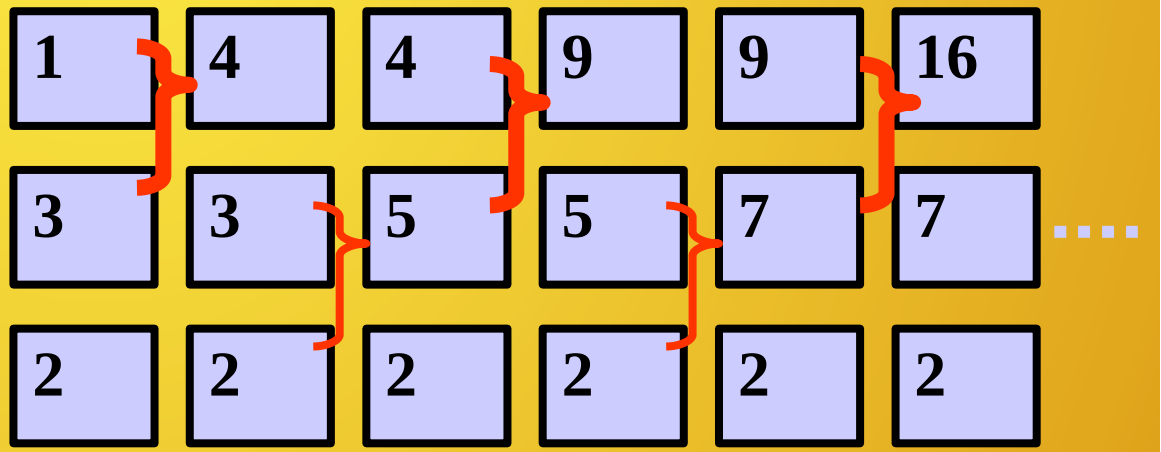
Babbage ugyancsak tudta, hogy egy

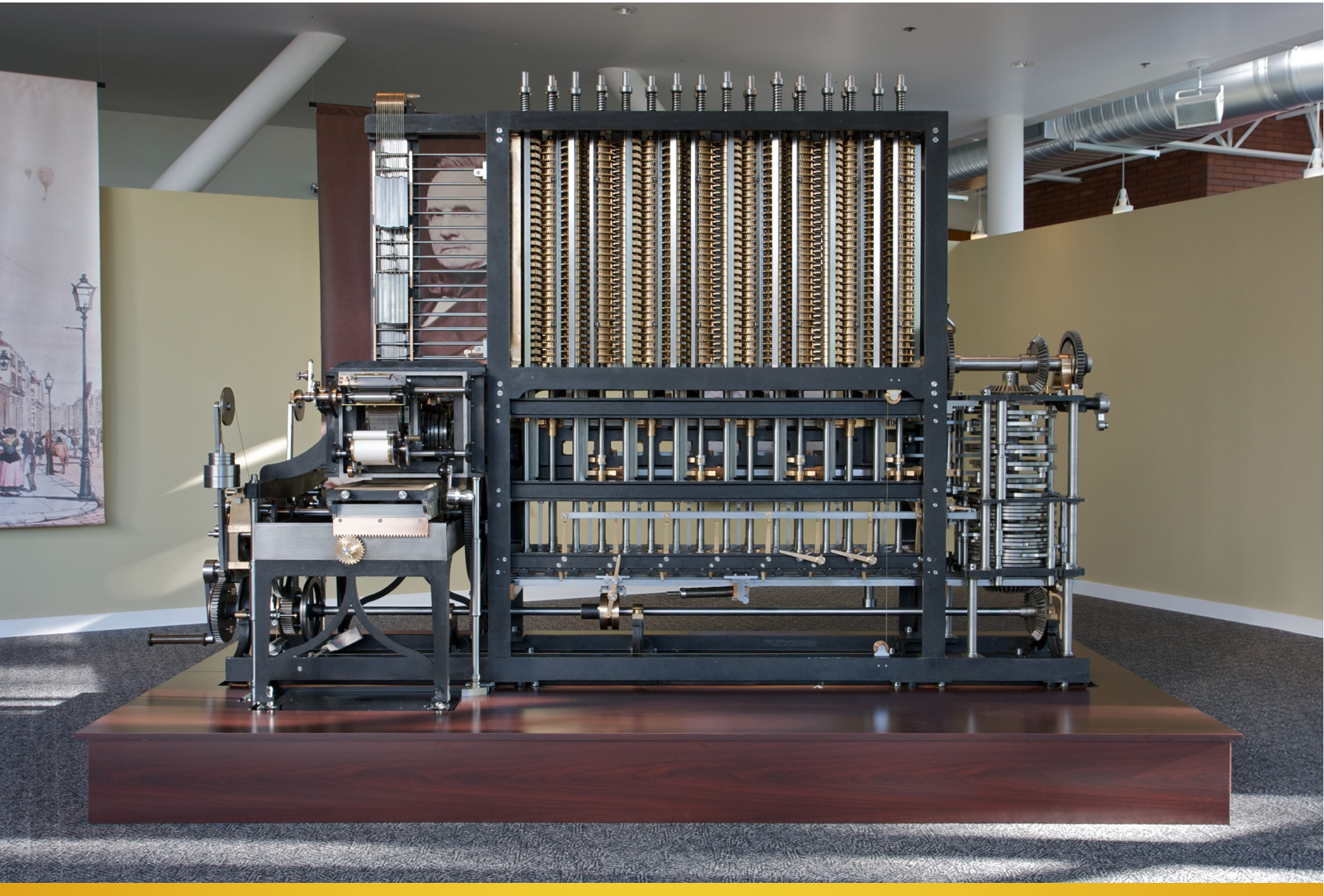
n-ed rendű polinóm n-edik differenciája konstans

Pl.: **y = x^2 + 2x + 1**

x	y	dy	d^2y
0	1		
		3	
1	4		2
		5	
2	9		2
		7	
3	16		2
		9	
4	25		2
		11	
5	36		








aX^2 + bX + c X= 0:N 3 digit pontos

A MEMÓRIA MEGJELENÉSE

Babbage megépített hat regiszteres differencia gépe maximum ötödfokú polinómok táblázatolását engedte meg.

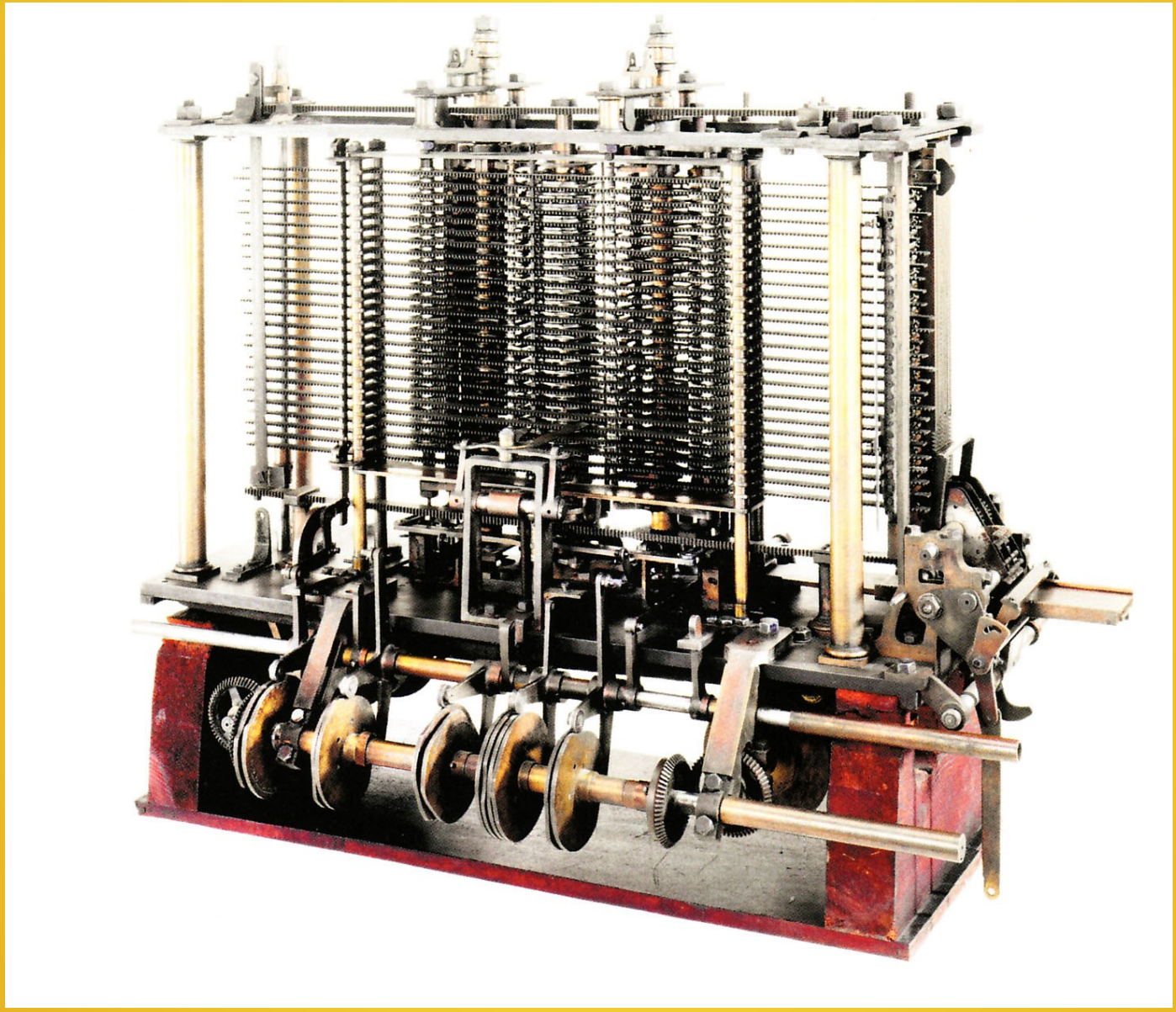
Bonyolultabb pl. trigonometrikus függvények megközelítéséhez nem elégséges egy ötödfokú hatványsor. 10, 20, 50 vagy tetszőleges n-ed fokú polinómokra lehet szükség. Ez ugyanennyi bonyolult fogaskerekes összeadómű megépítését tenné szükségessé.

És itt jött Babbage korszaknyitó ötlete: Használjunk csak egyetlenegy -- felépítésében a szűkséges fogaskerékrendszer miatt bonyolult -- összeadó-művet, vagy általában aritmetikai egységet, s n darab közönséges keréktárcsákból álló tároló regisztert. Ez lesz a MEMÓRIA.

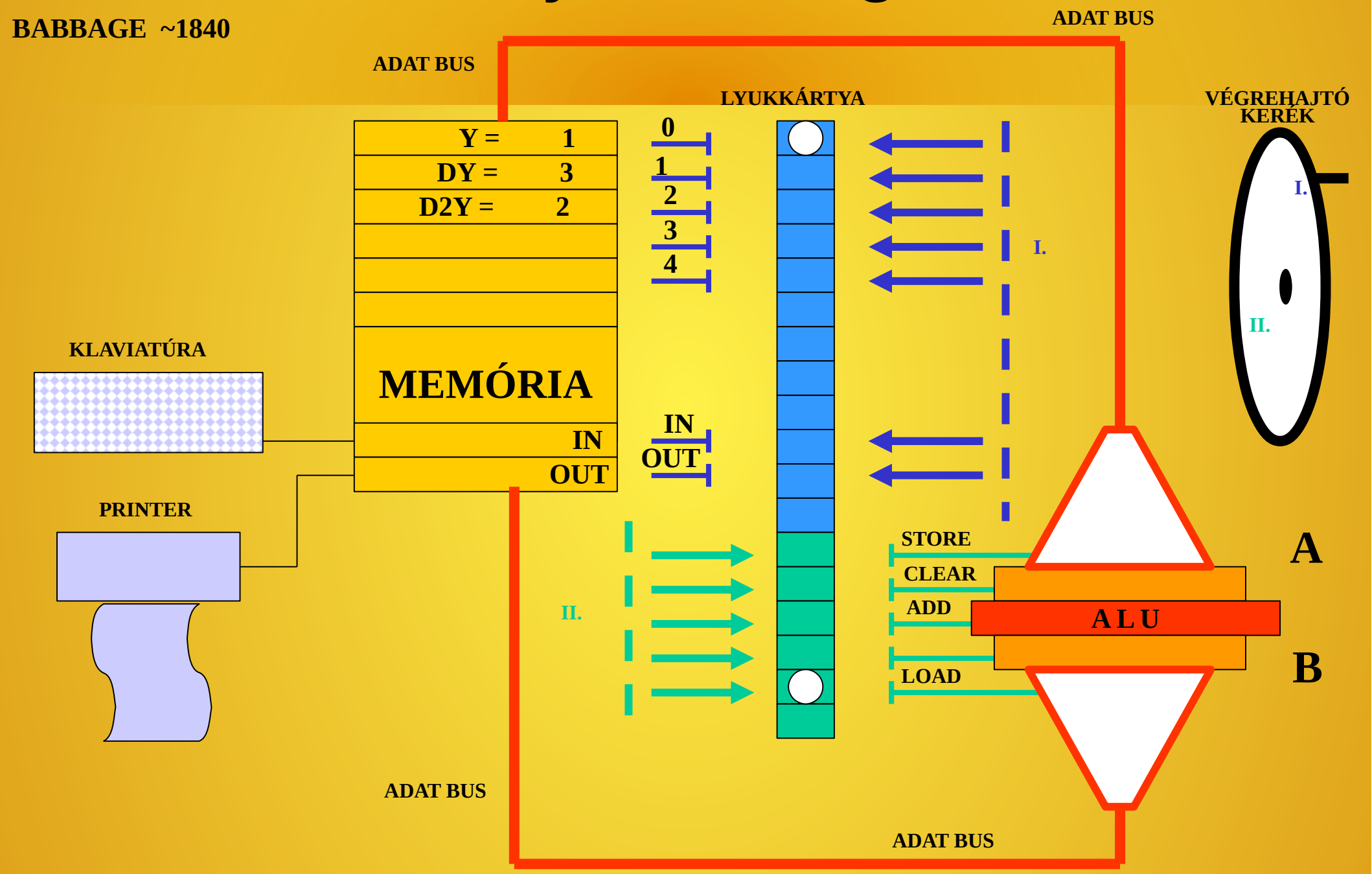
Meg kell oldani, hogy a kezdő és átmeneti értékeket tároló memória-regiszterek tartalma -- valamilyen módon megvalósítandó adatátvitel révén -- cserélhető legyen az aritmetikai egység A és B regisztereinek tartalmával.

Az adatátvitelek sorrendjét egy lyukkártyán tárolt program alapján működő vezérlőmű vezérelné. És ez lett a később megtáltosodó gondolatok magja.

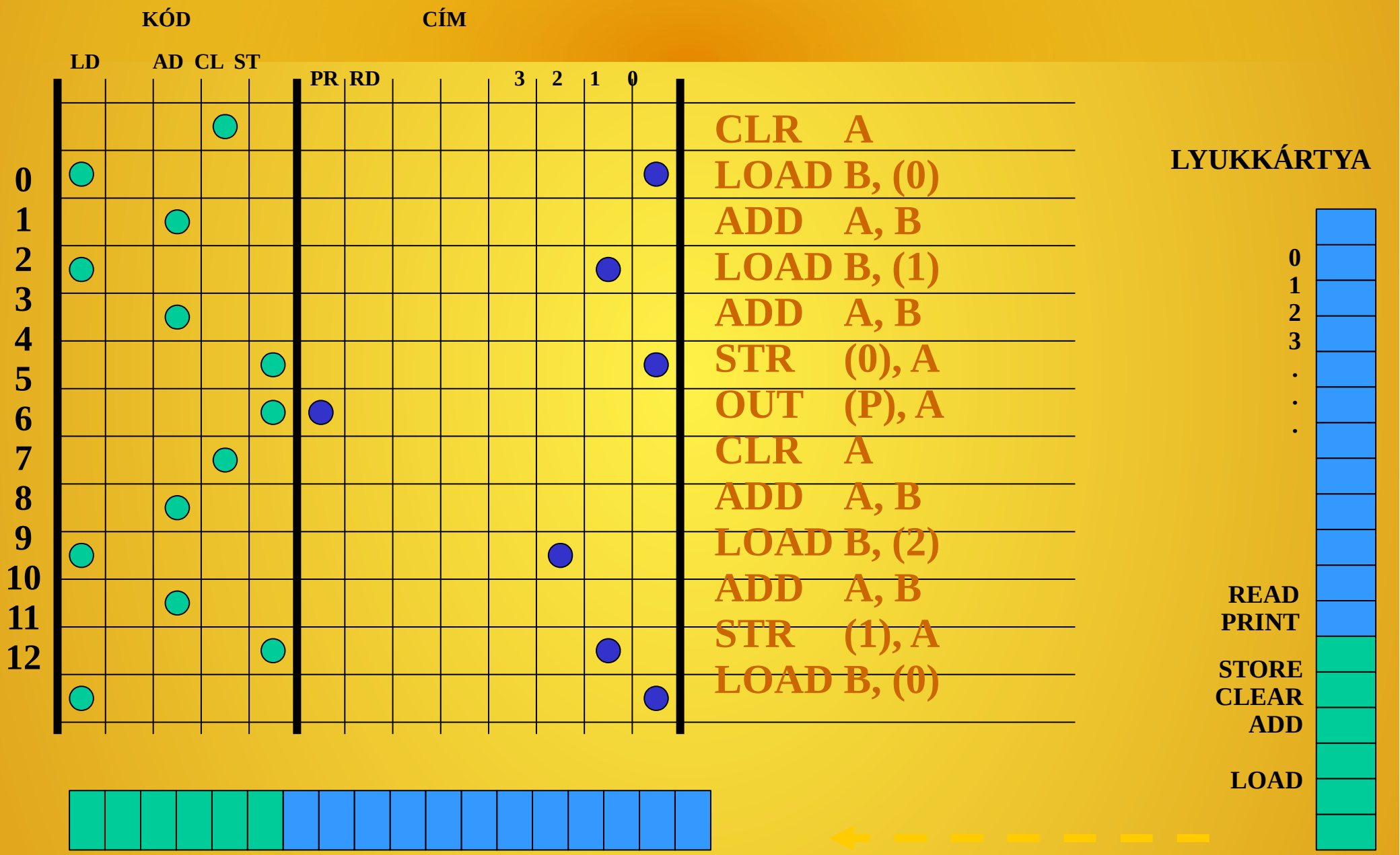
Analytical Engine



Analytical Engine



GÉPI és ASSEMBLY KÓD



KETTES SZÁMRENDSZER

A tizes számrendszerben működő aritmetikai egység és az ugyancsak tizes számrendszerű memóriarekeszek közötti

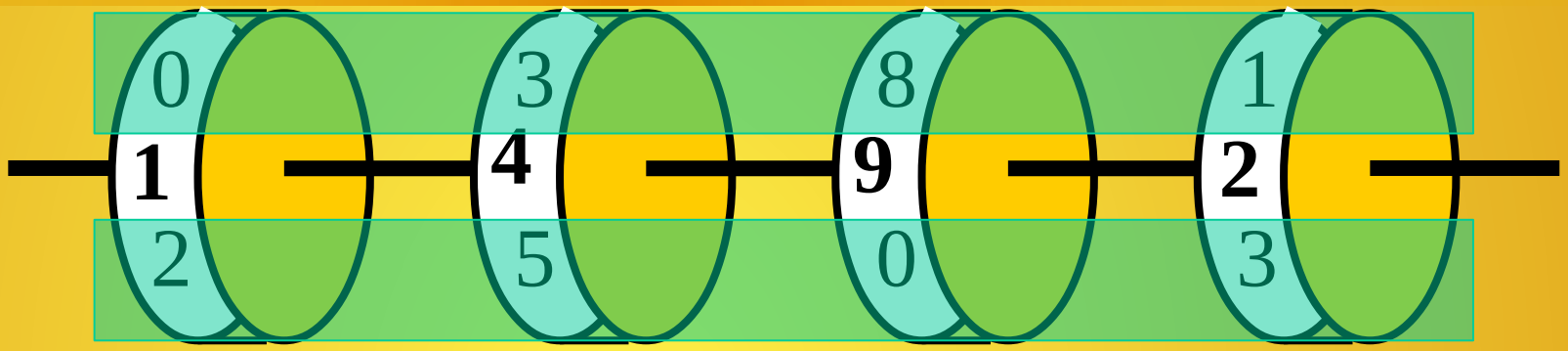
ADATÁTVITEL

nem volt megvalósítható a kor technológiai szinvonalán Babbage minden erőfeszítései ellenére sem.

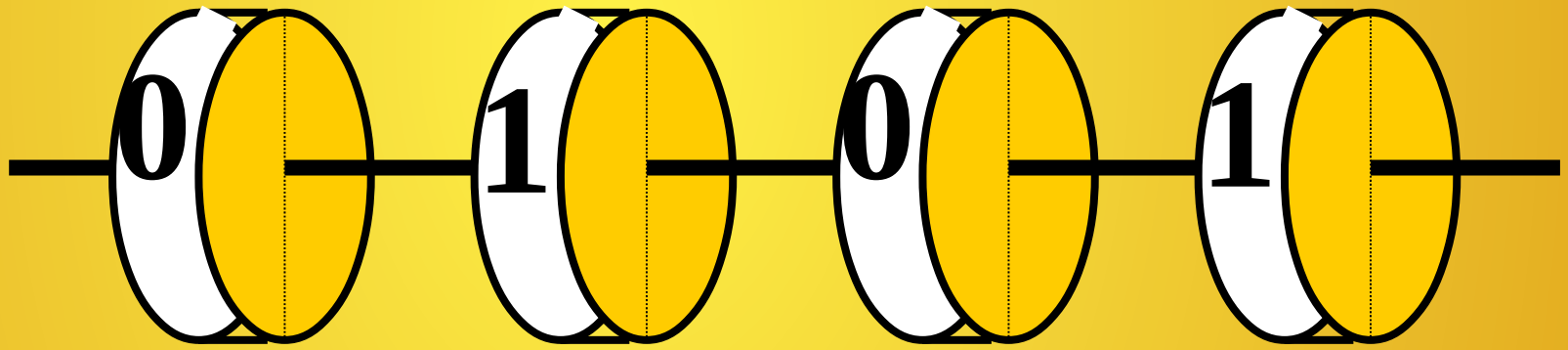
A minimális számjegyű, csak két számjegyet tartalmazó kerekek felvetése villantott fel a megoldható adatátvitelre némi reményt, de csak 100 év késéssel és kerekek nélkül.

REGISZTEREK

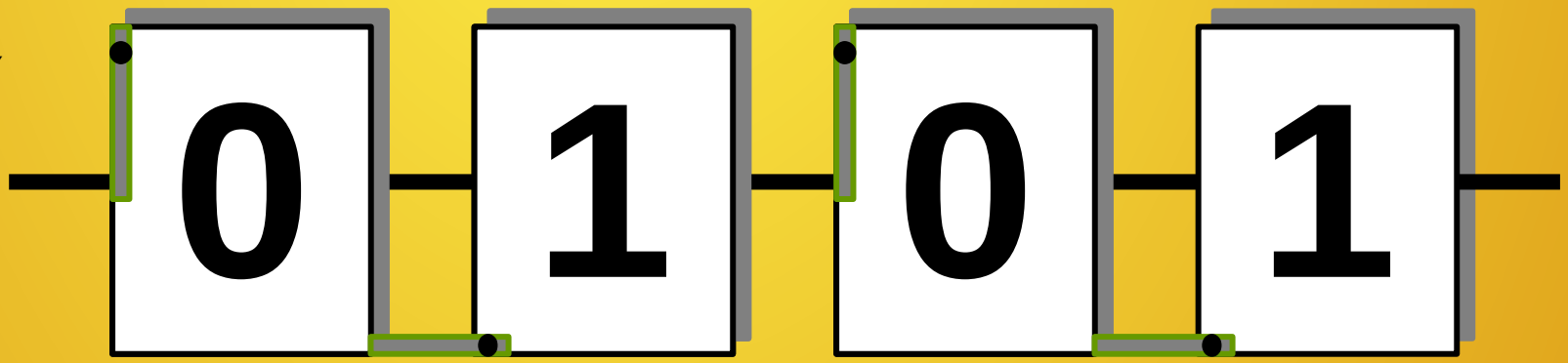
DECIMÁLIS
(TIZES)

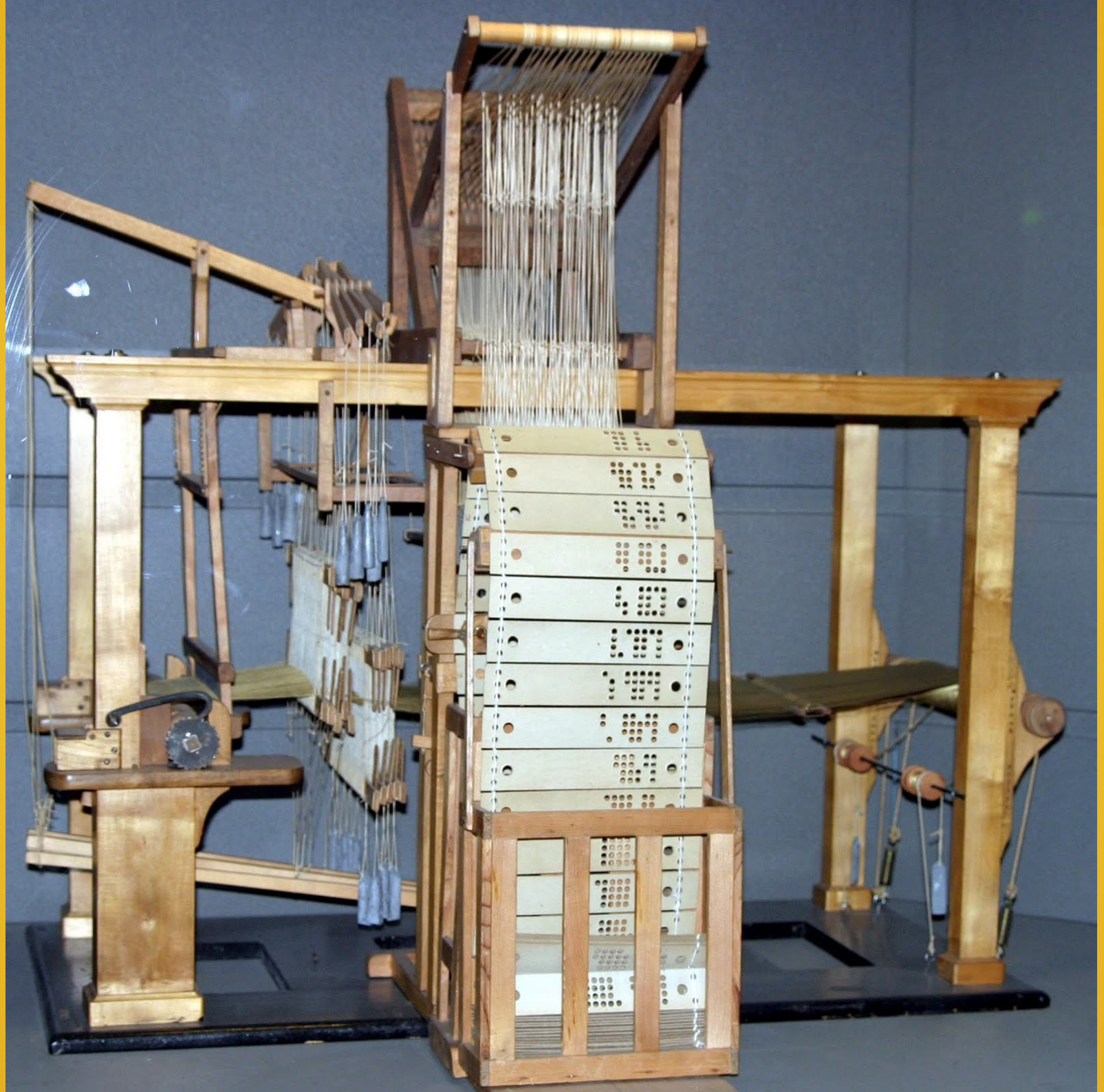


BINÁRIS
(KETTES)



HA BINÁRIS, ÚGY
DOMINÓKBÓL
IS LEHETNE



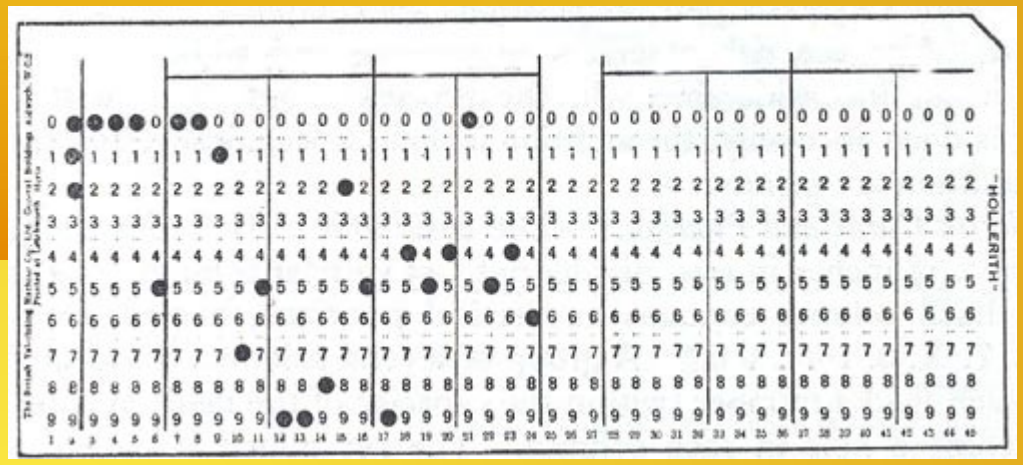


ÍRHATÓ - OLVASHATÓ REGISZTEREK

Két értéket (bináris jegyet, bitet) tároló (elektromosan írható és olvasható) eszközt először relékkel, majd elektronikusan, úgy nevezett FLIP-FLOP-okkal tudtak megvalósítani. A kerekek alkalmazása helyett ezekkel a működési sebesség jelentősen megugorhatott.

1944-46 között NEUMANN János elemezte, hogy ilyen ígéretes sebességek mellett milyen kell legyen egy számítógép méltó architektúrája. Megszületett a tárolt programmozású COMPUTER gondolata.
A több száz millió PC is ‘von Neumann computer’.

Herman Hollerith



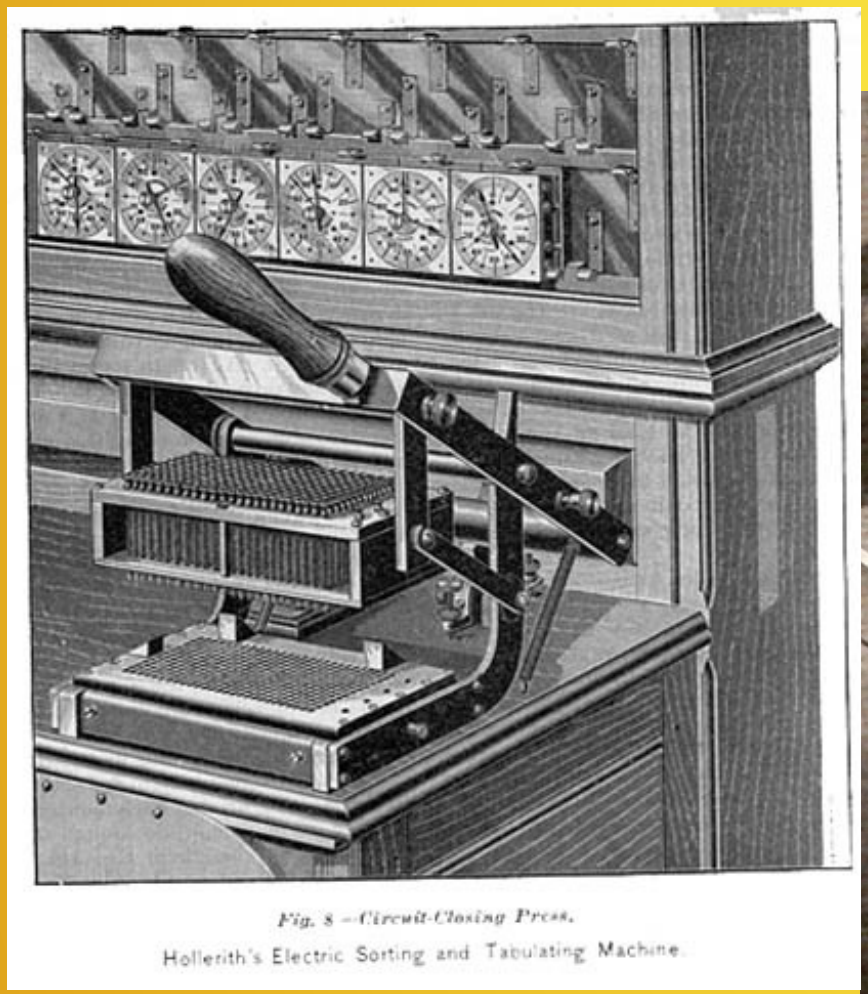
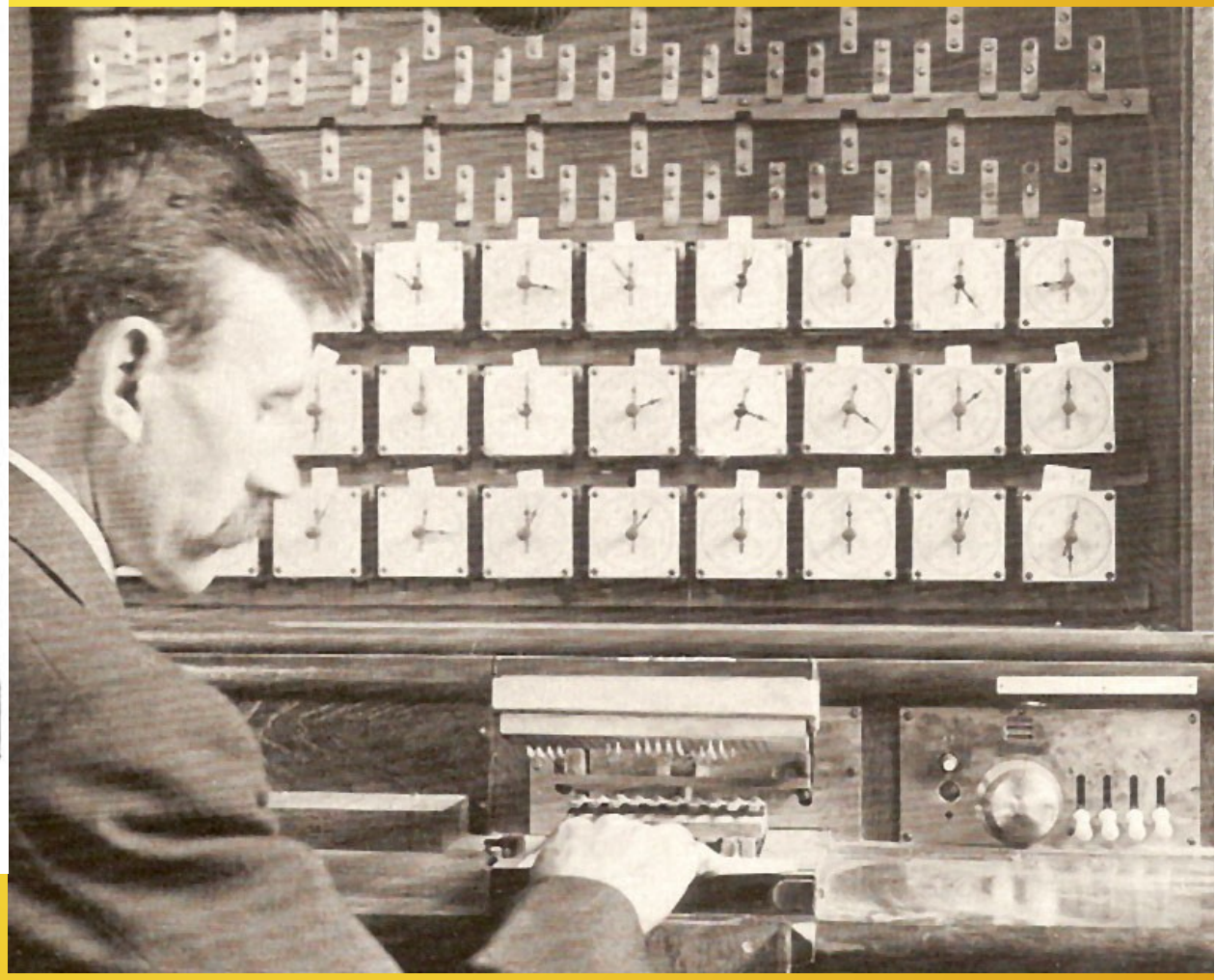


Fig. 8.—Circuit-Closing Press.

Hollerith's Electric Sorting and Tabulating Machine.



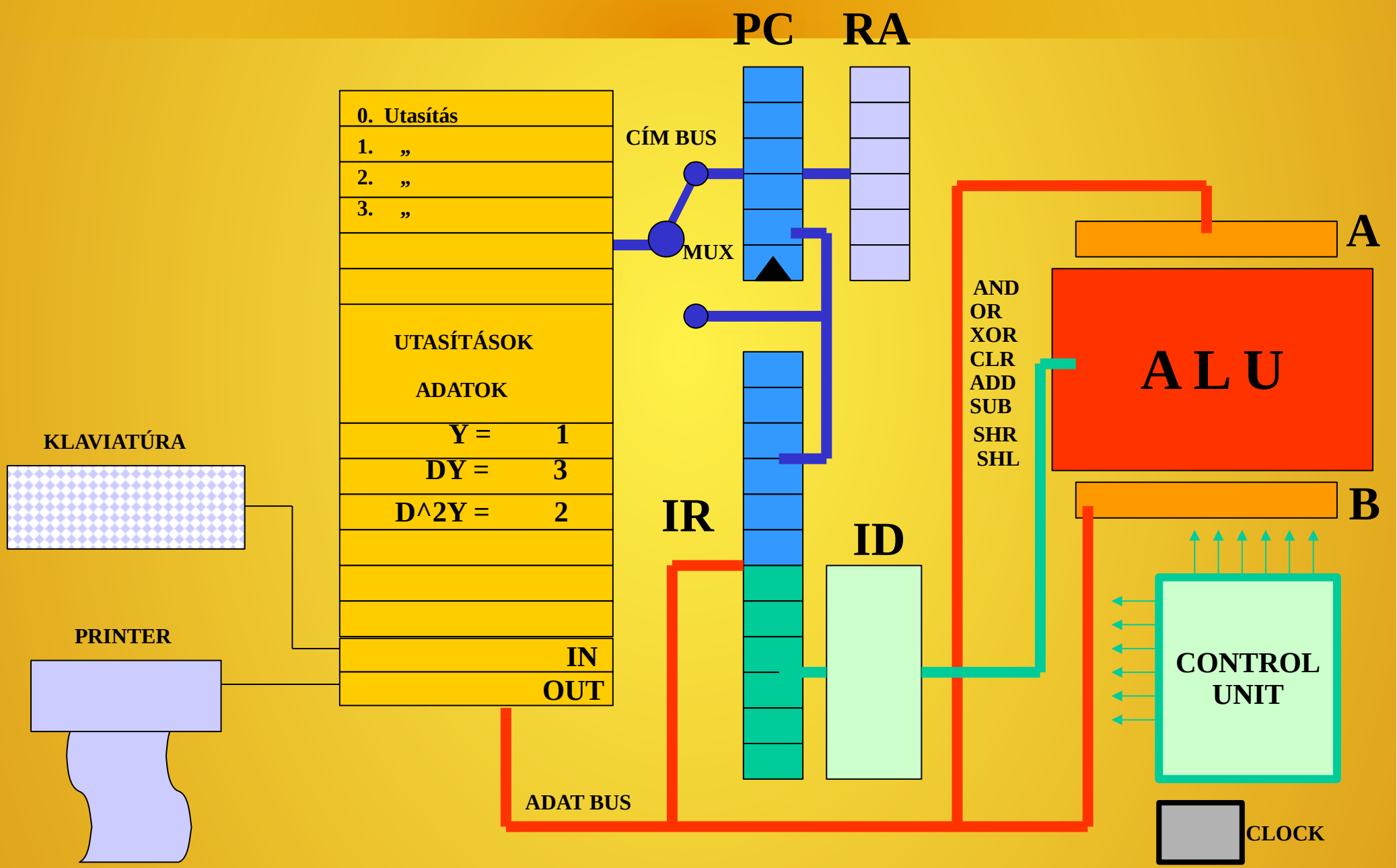
NEUMANN János

Budapest 1903 - Princeton (USA) 1957

Von NEUMANN COMPUTER

NEUMANN ~1944

MEMÓRIA **PROCESSZOR**

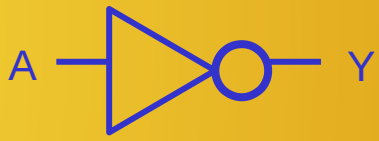


A LOGIKA ELEMEI

A logikai változók lehetséges értékei: **0 , 1**

Ezeket 0 V illetve +5 V feszültségszinttel fogjuk reprezentálni

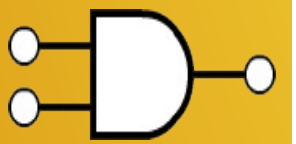
Egyváltozós logikai művelet a **NEGÁLÁS**: **Y = A’**

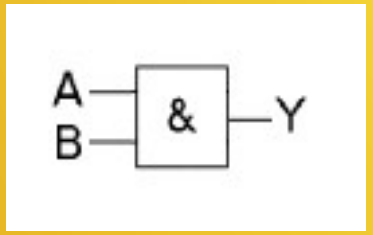


Kétváltozós logikai művelet az

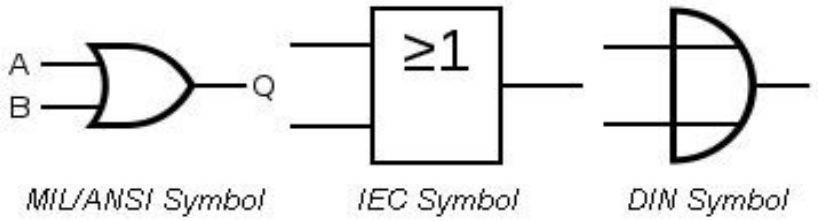
ÉS és **VAGY**

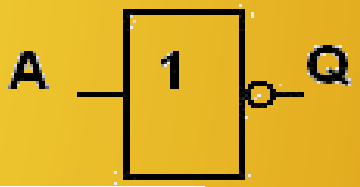
Y = AB



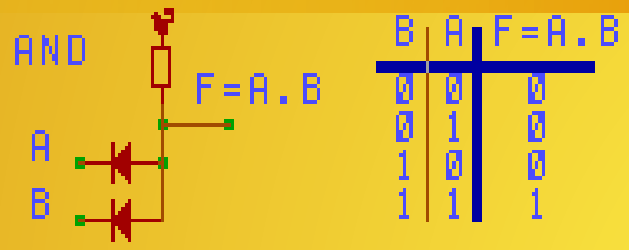


Y = A+B

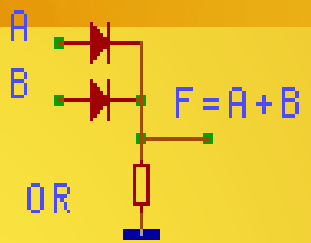




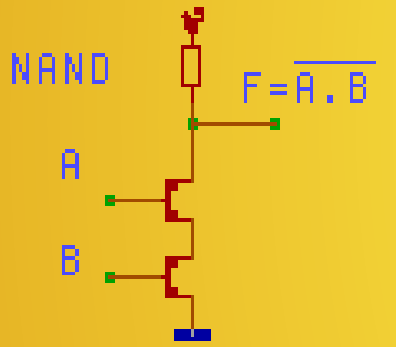
LOGIKAI KAPUK



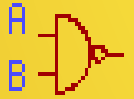
B	A	F=A.B
0	0	0
0	1	0
1	0	0
1	1	1

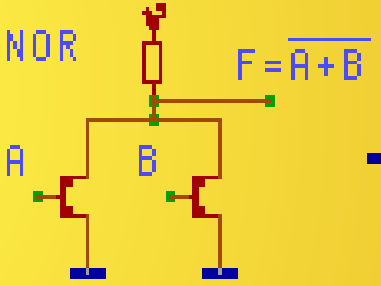


B	A	F=A+B
0	0	0
0	1	1
1	0	1
1	1	1

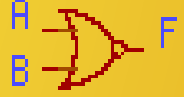


B	A	F=$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0





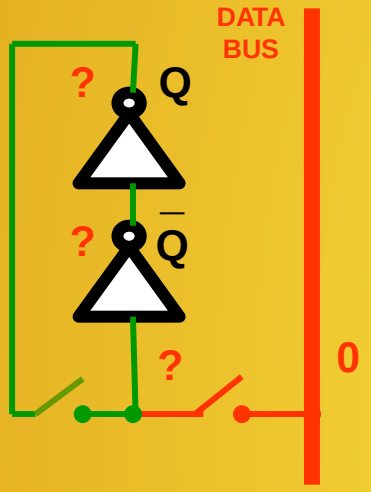
B	A	F=$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

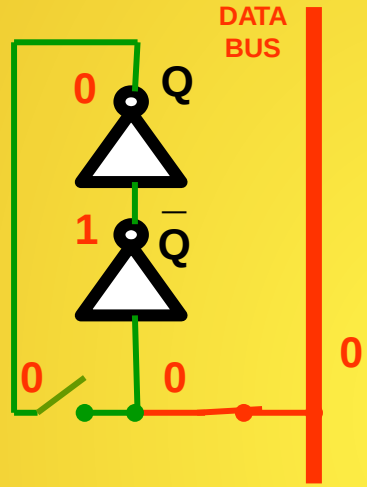


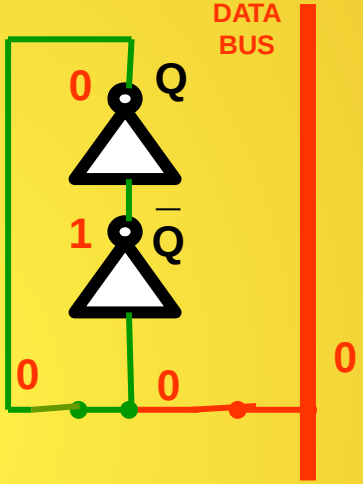
All possible two-variable logic (or Boolean) functions

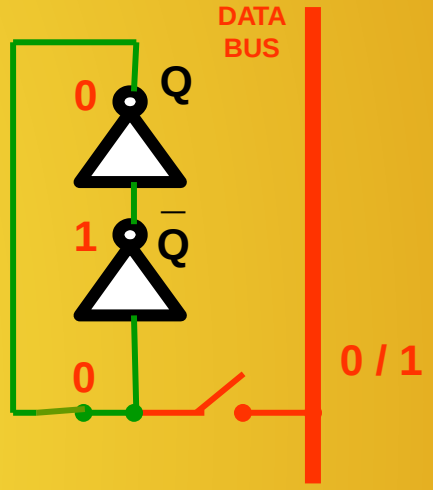
which could be implemented by two-input logic gates

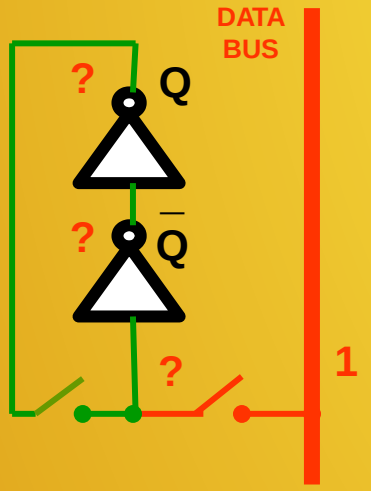
B	A	f_0	NOR	f_2	f_3	f_4	f_5	XOR	NAND	AND	EQU	f_{10}	f_{11}	f_{12}	f_{13}	OR	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

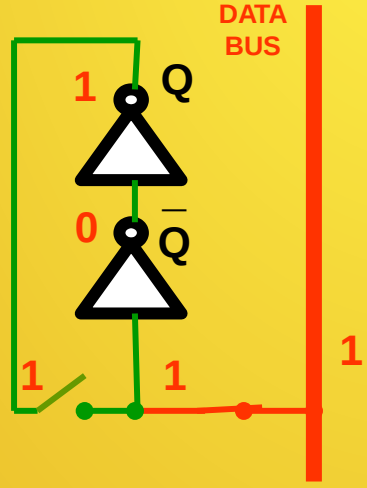

DATA
BUS
?
Q
?
Q
?
0

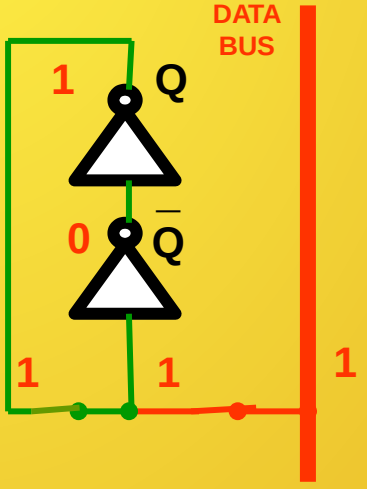

DATA
BUS
0
Q
1
Q
0
0
0

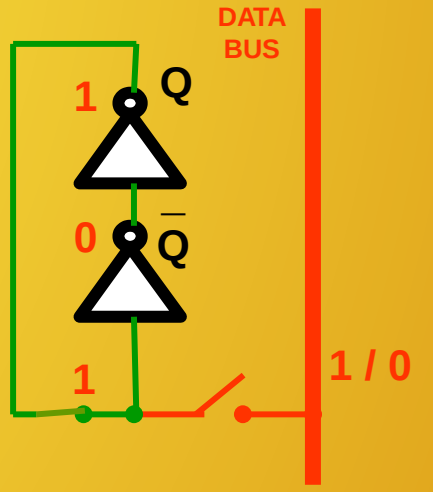

DATA
BUS
0
Q
1
Q
0
0
0


DATA
BUS
0
Q
1
Q
0
0 / 1

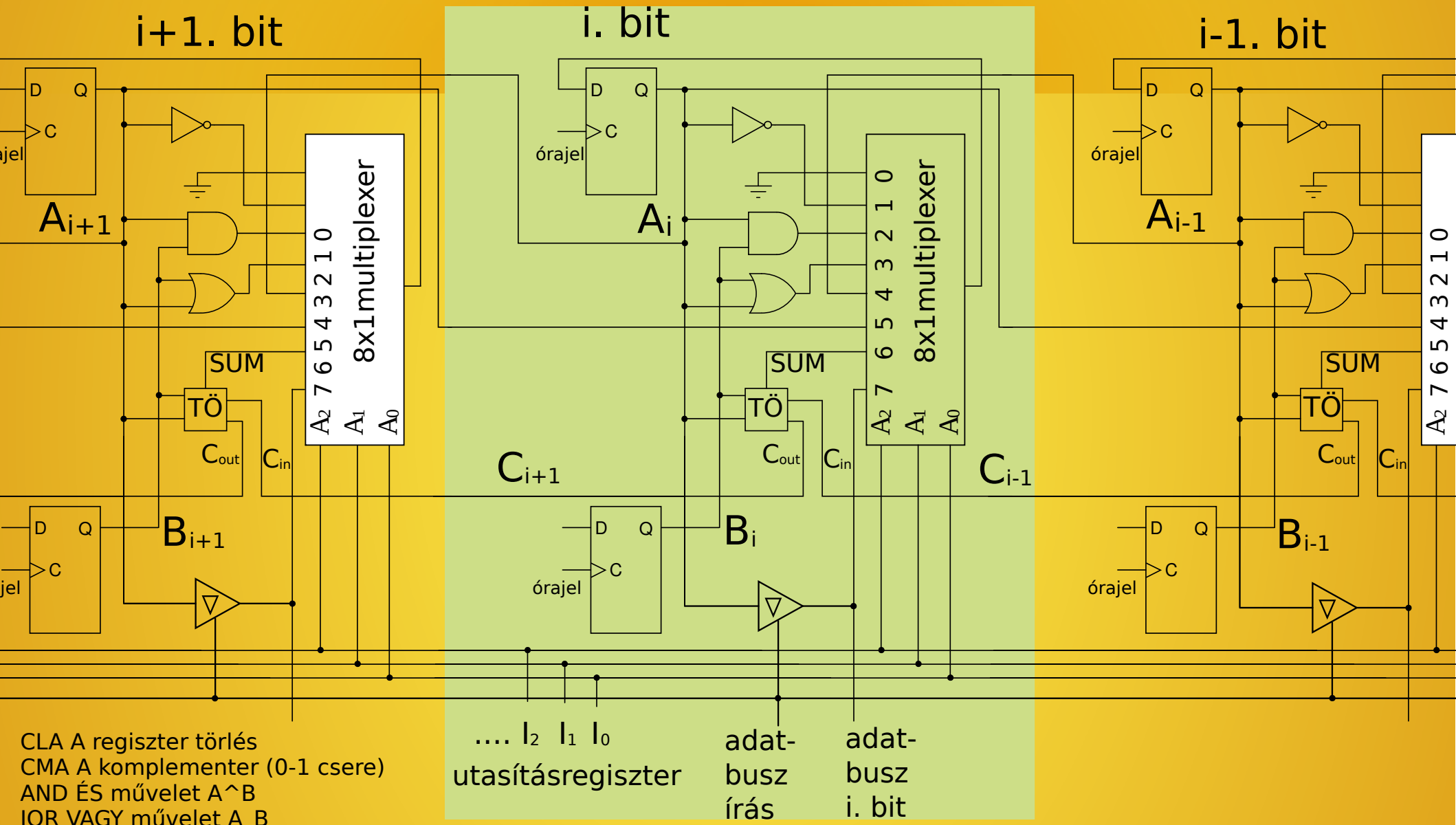

DATA
BUS
?
Q
?
Q
?
1


DATA
BUS
1
Q
0
Q
1
1
1


DATA
BUS
1
Q
0
Q
1
1
1


DATA
BUS
1
Q
0
Q
1
1 / 0

ARITMETIKAI-LOGIKAI EGYSÉG

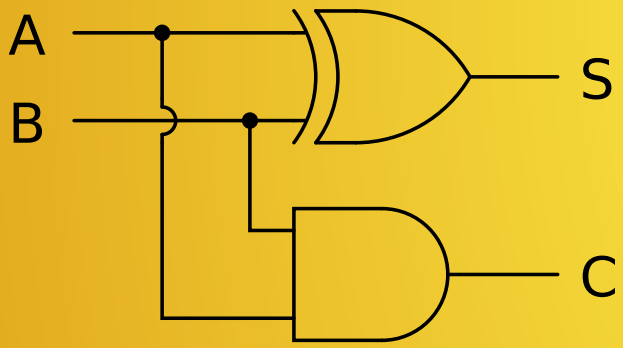


CLA A regiszter törlés
CMA A komplementer (0-1 csere)
AND ÉS művelet A^B
IOR VAGY művelet A_B
RAL A bitenkénti eltolás balra (szorzás 2-vel)
RAR A bitenkénti eltolás jobbra (osztás 2-vel)
ADD összeadás A + B
LD olvasás az adatbuszról az A regiszterbe

ÖSSZEADÓ

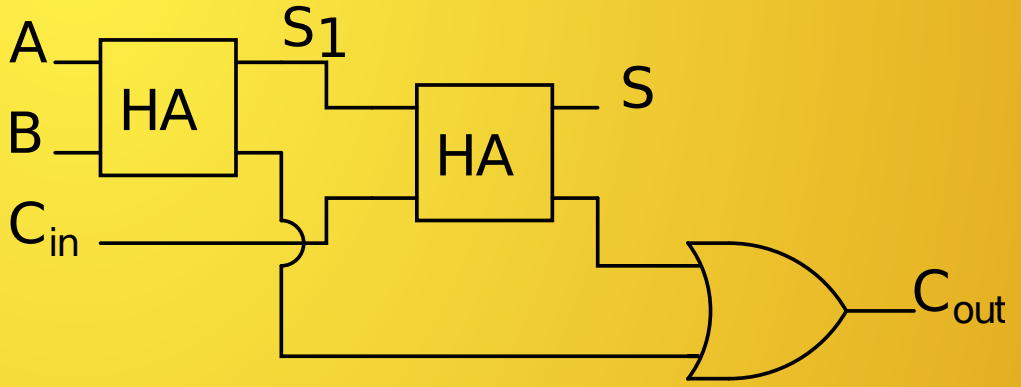
Félösszeadó, HA:

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

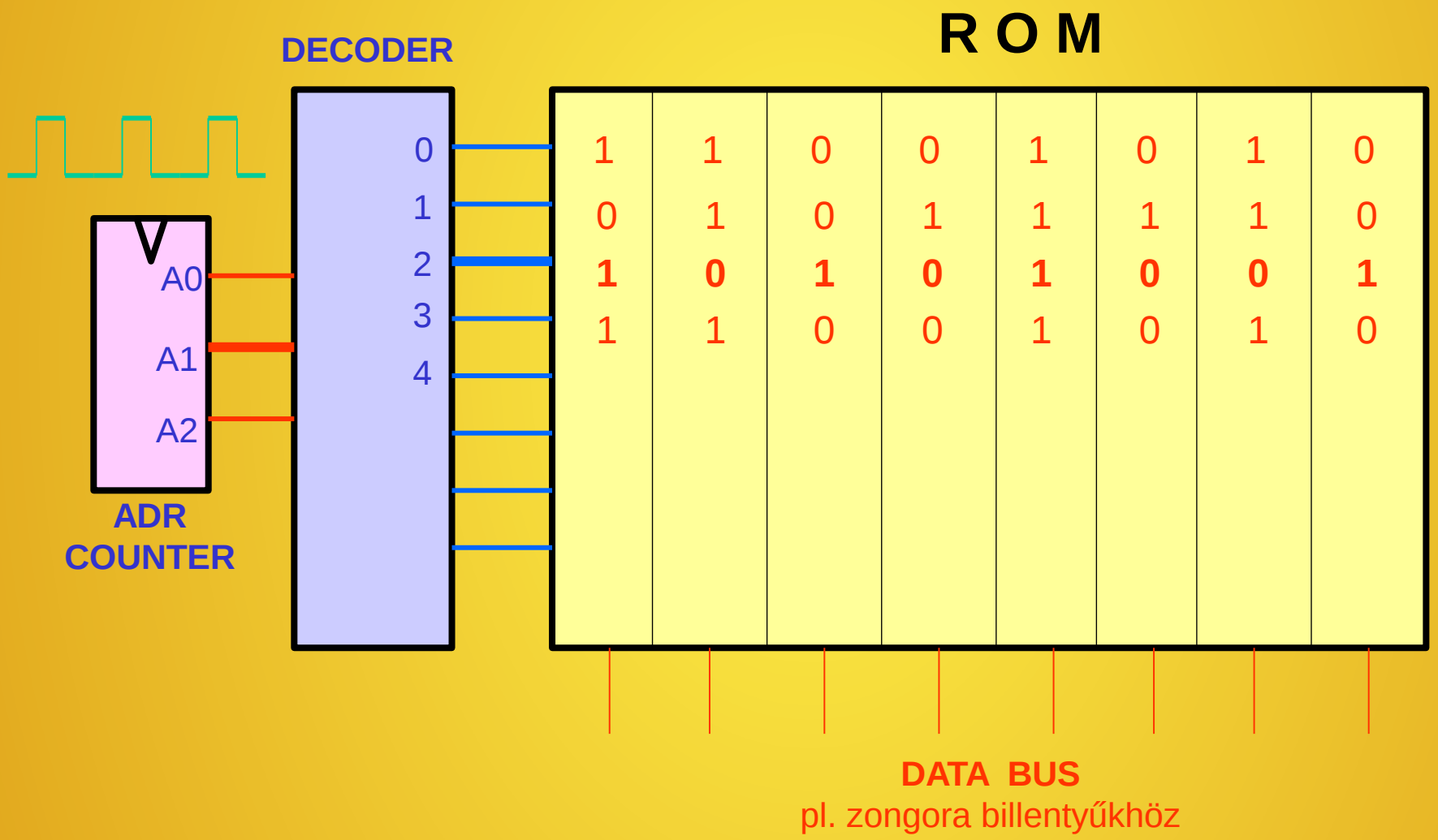


Teljes összeadó:

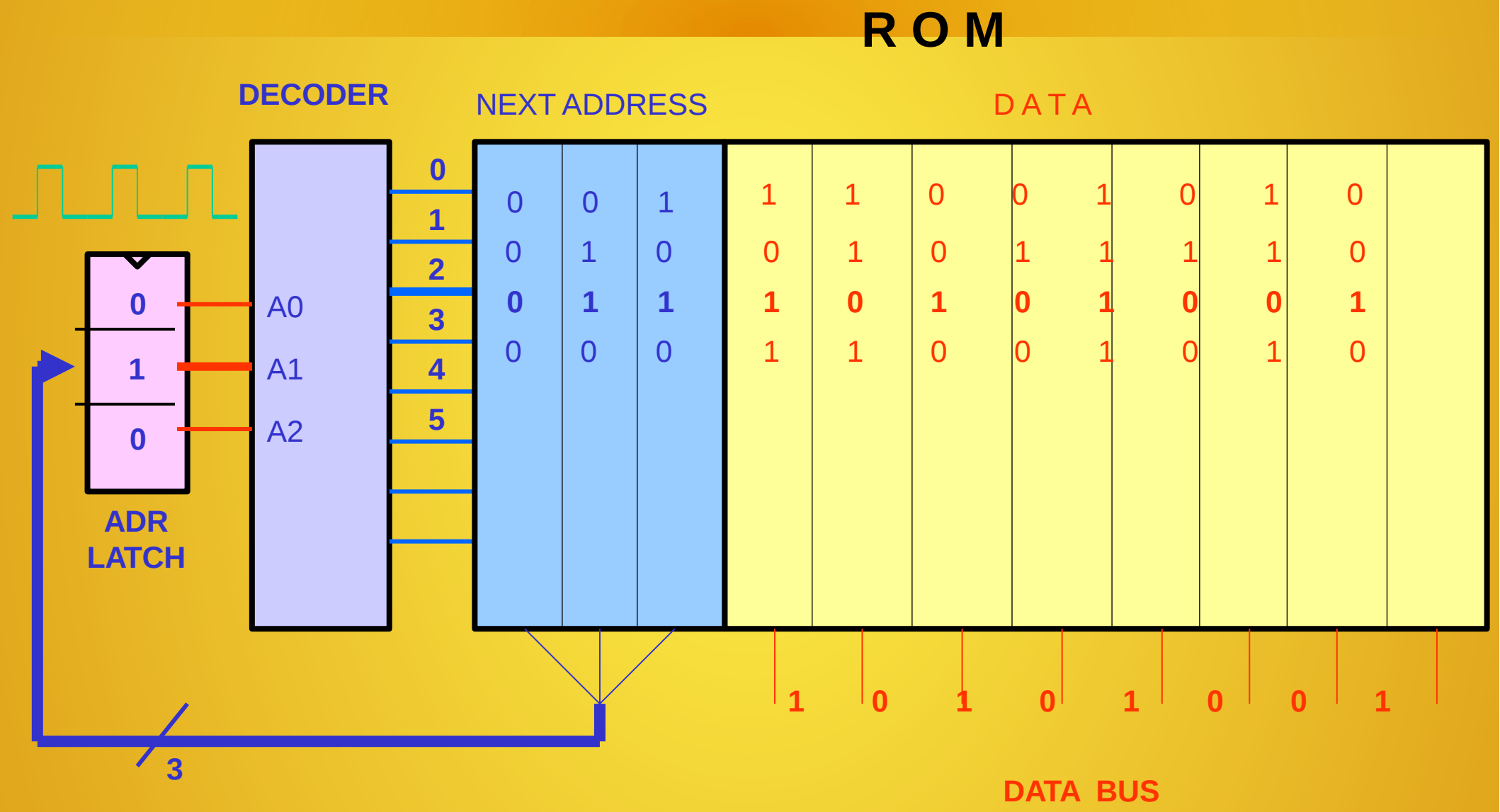
c_{i-1}	a_i	b_i	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



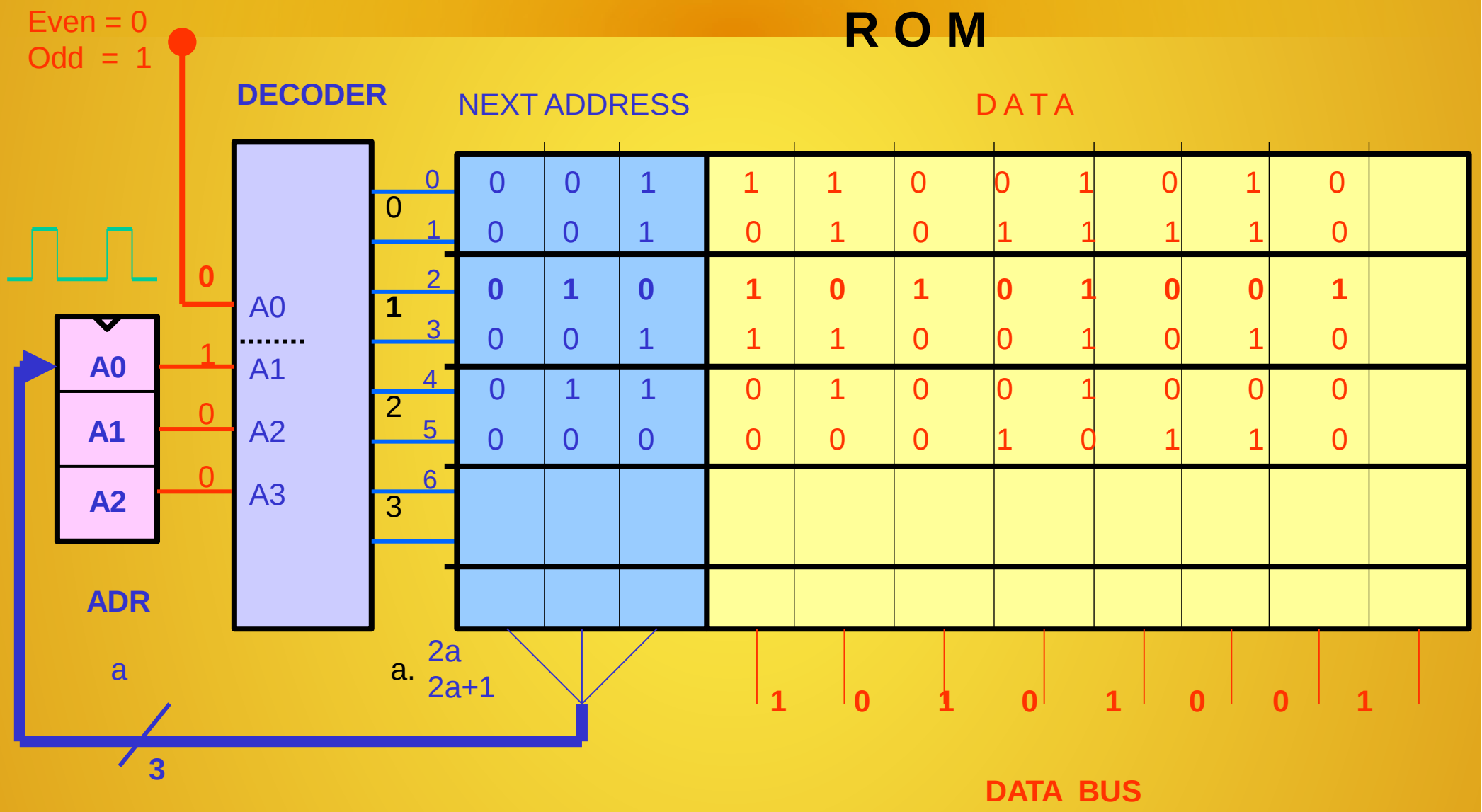
ELEKTRONIKUS VERKLI



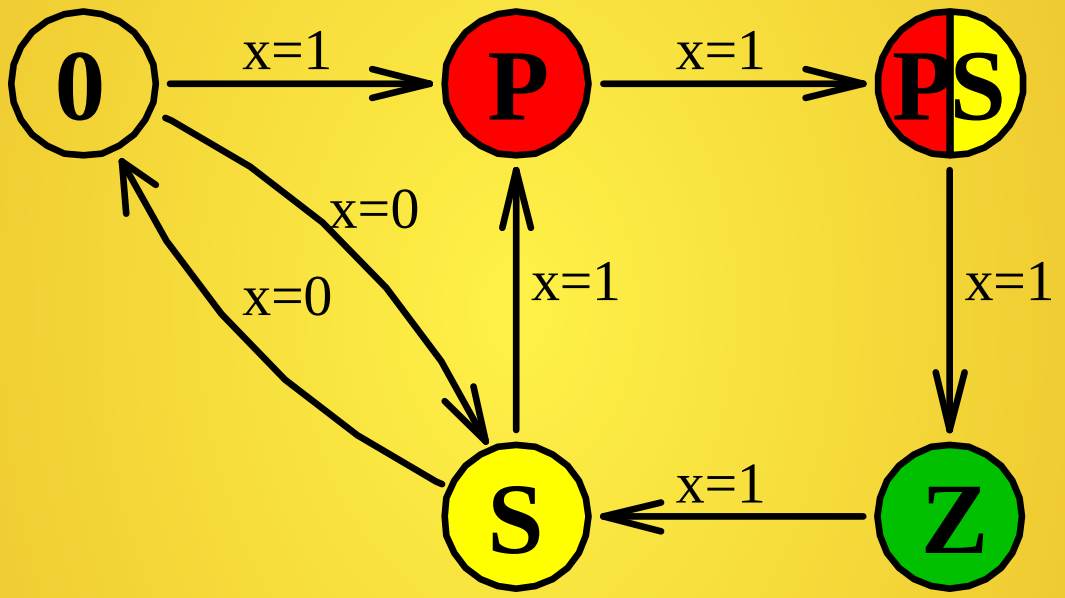
FLEXIBILISEBB VERKLI



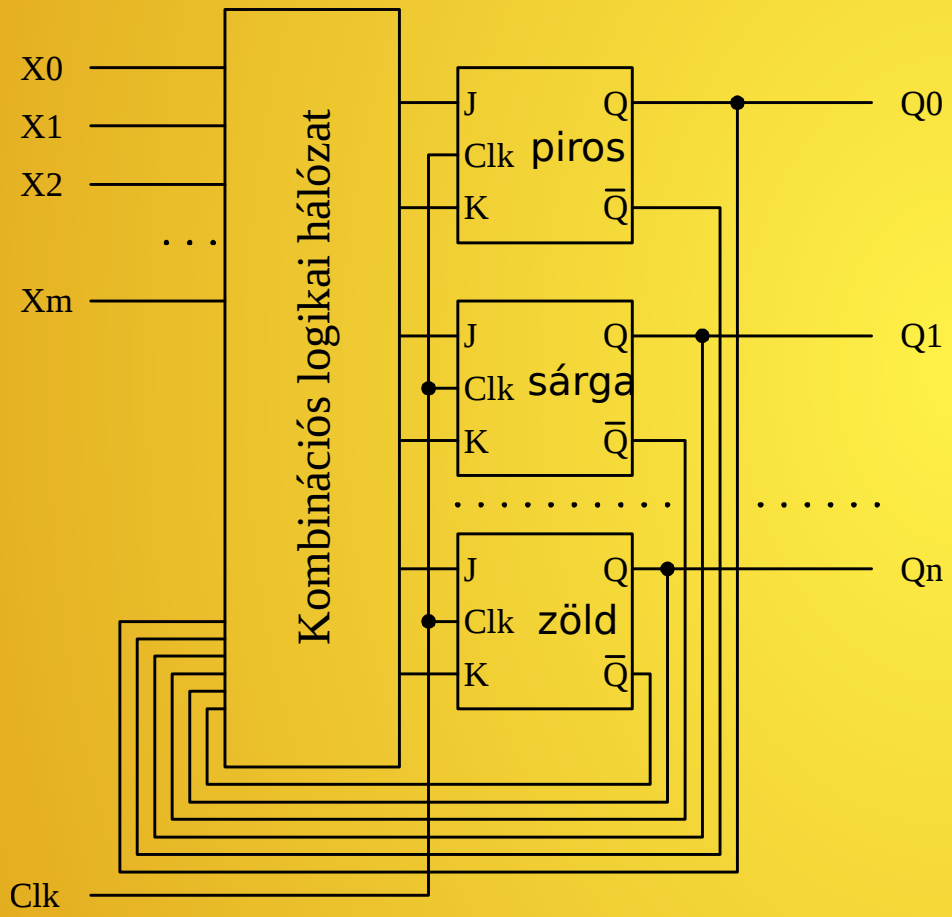
MÉG FLEXIBILISEBB VERKLI

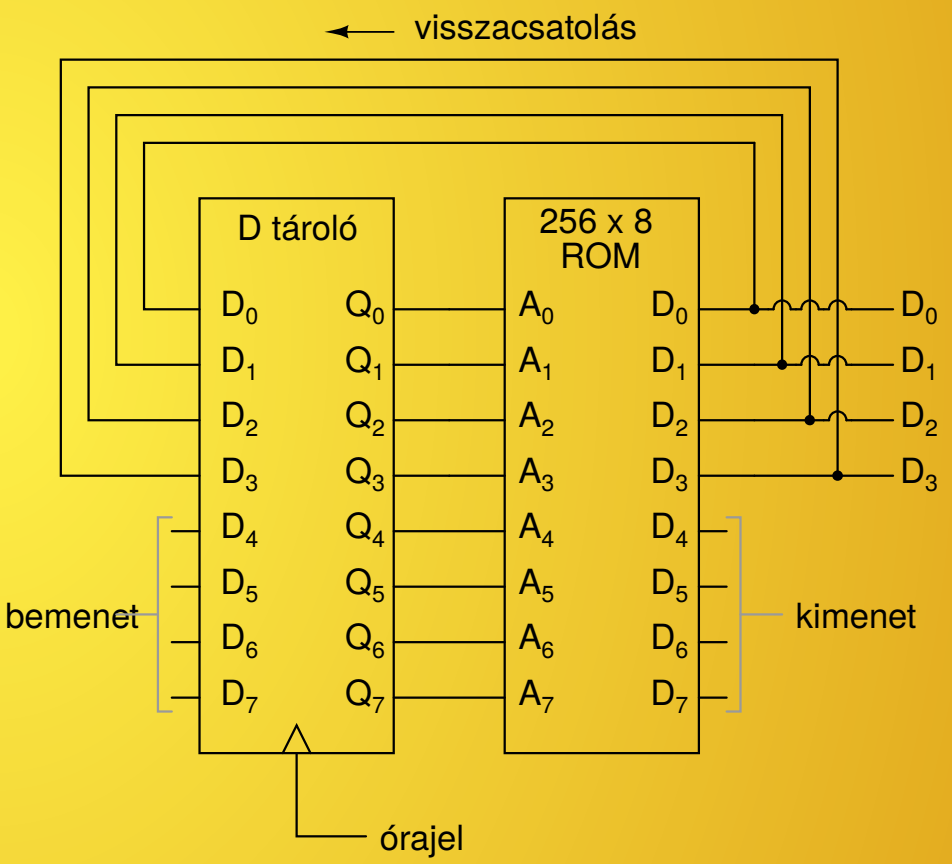


Közlekedési lámpa



Véges állapotú automata / Algorithmic State Machine

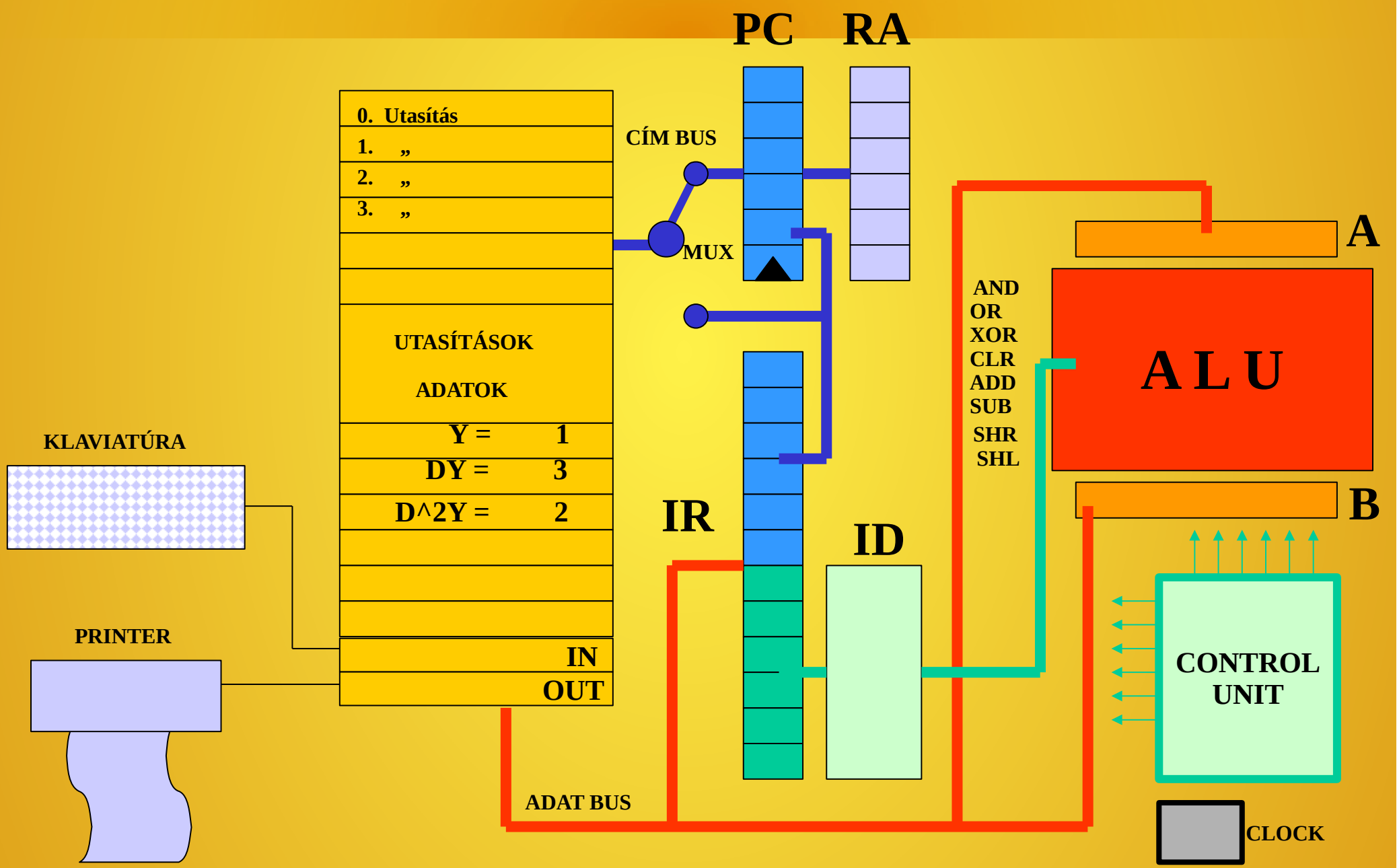




Von NEUMANN COMPUTER

NEUMANN ~1944

MEMÓRIA **PROCESSZOR**



INPUT - OUTPUT UTASÍTÁSOK

ASSEMBLY Language

```
MOV     DX,     300H        ;Move Port Address into DX
IN      AL,     DX          ;Input from Port into AX

MOV     DX,     300H        ;Move Port Address into DX
MOV     AL,     55H         ;Move DATA (55H) into AL
OUT     DX,     AL          ;Output DATA to Port
```

BASIC

```
PA = &H300                  ;Port Address Assignment
DATA = INP ( PA )           ;Input from Port PA into DATA
OUT PA, DATA                ;Output DATA to Port PA
```

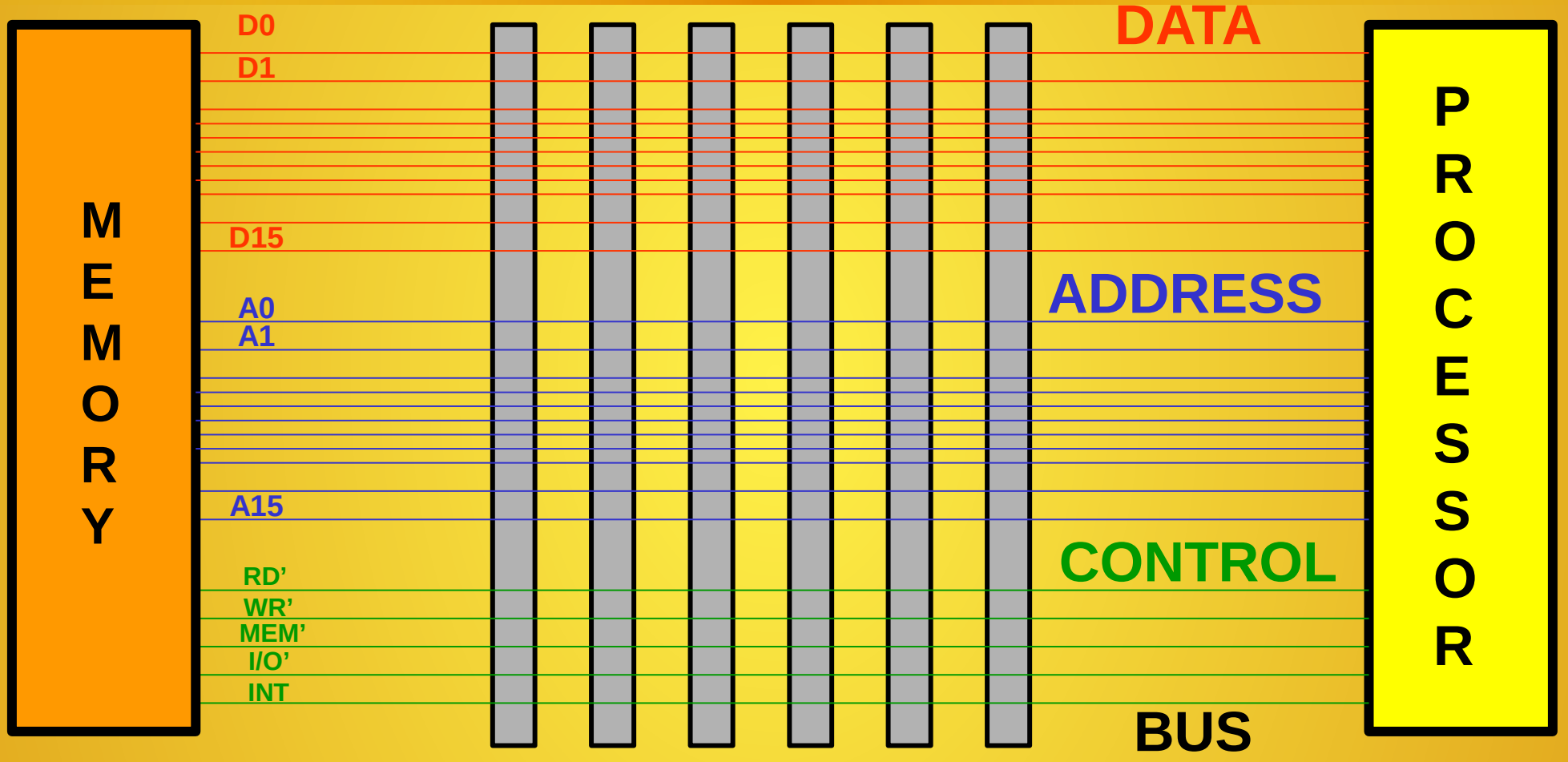
PASCAL

```
PA := $300;                 {Port Address assignment}
DATA := PORT [PA];          {Input from Port into DATA}
PORT [PA] := $55;           {Output DATA to Port PA}
```

C

```
pa = 0x300;                 /*Port Address assignment*/
indata = inportb (pa);      /*Input from Port into indata*/
outportb (pa,outdata0;      /*Ouput outdata to Port pa*/
```

ADAT, CÍM és CONTROL BUS



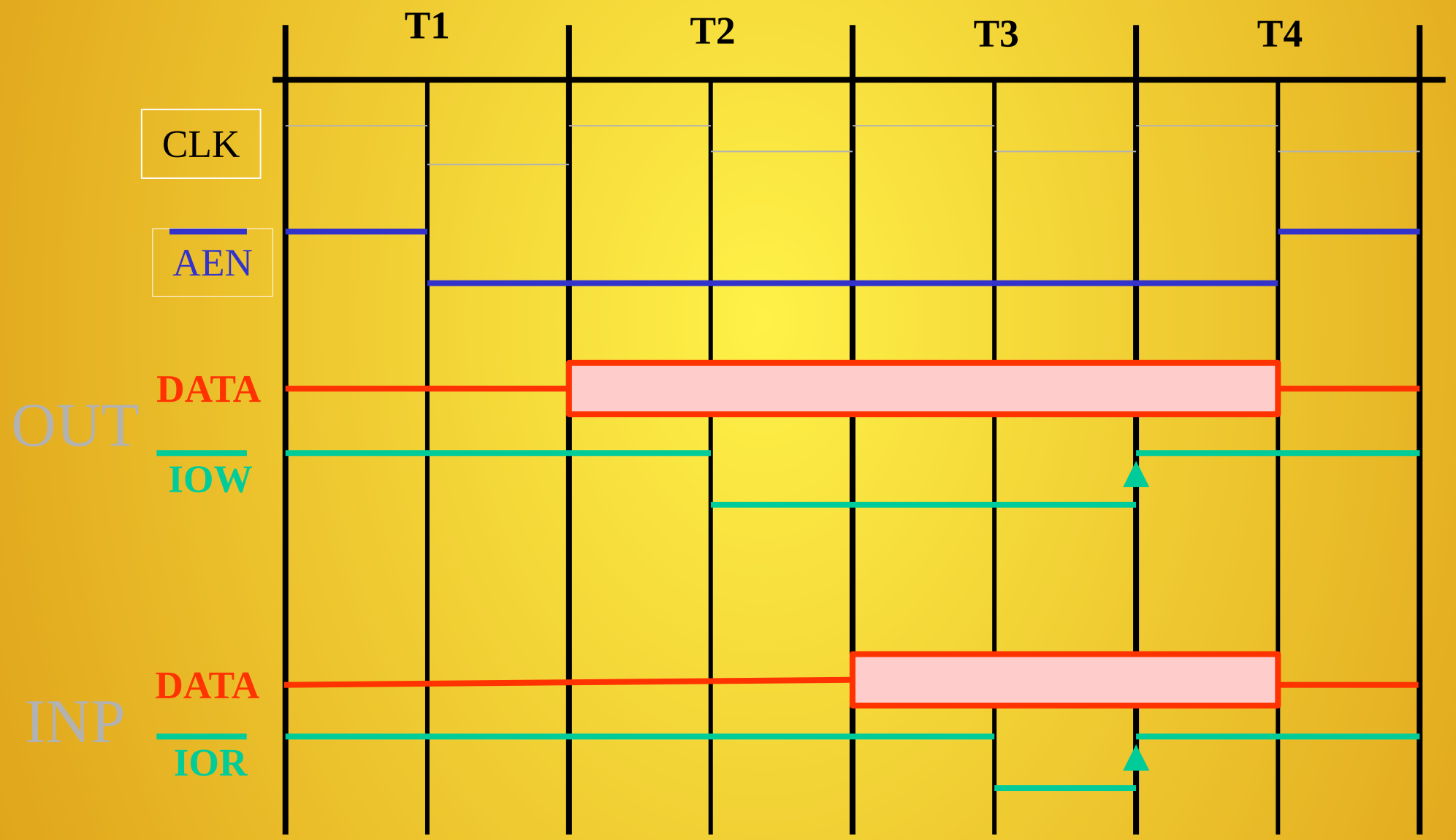
BUS CONNECTORS
for I / O interfaces

Ezeket nekünk kell csinálni

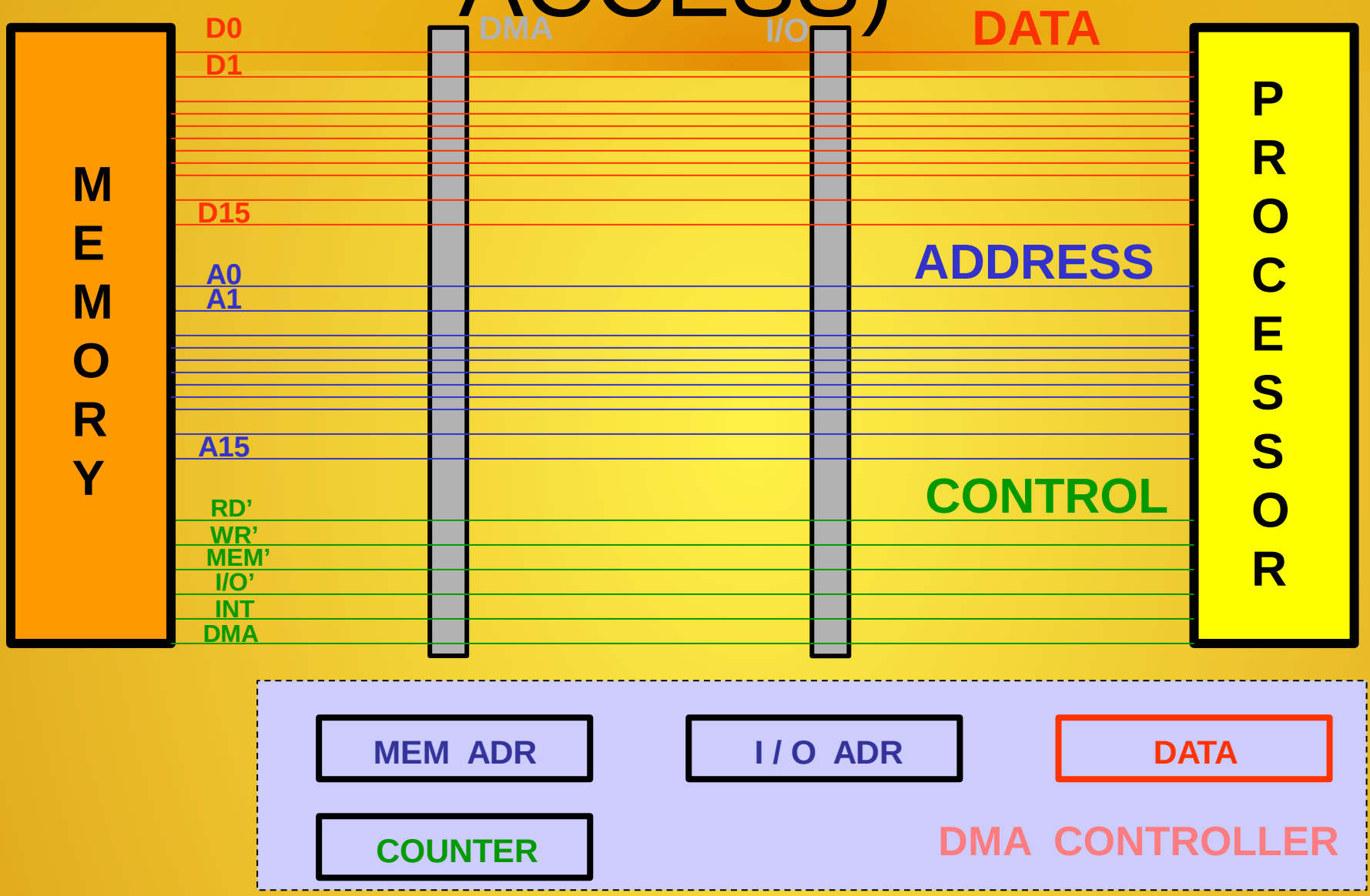
Interfészek:

KEYBOARD | DISPLAY | DISK | PRINTER | SERIAL | SPECIAL

I/O CIKLUS IDŐDIAGRAMMJA



DMA (DIRECT MEMORY ACCESS)



Intel Pentium 5 Prescott

Trace Cache Access, next Address Predict

Trace Cache Branch Prediction Table (BTB), 1024 entries.

Return Stacks (4 x16 entries)

Trace Cache next IP's (4x)

Instruction Decoder

Up to 4 decoded uOps/cycle out. (from max. one x86 instr/cycle) Instructions with more than four are handled by Micro Sequencer

Raw Instruction Bytes in

Data TLB, 64 entry fully associative, between threads dual ported (for loads and stores)

Front End Branch Prediction Tables (BTB), shared, 4096 entries in total

Instruction TLB's 128 entry, fully associative for 4k and 4M pages. In: Virtual address [47:12] Out: Physical address [39:12] + 2 page level bits

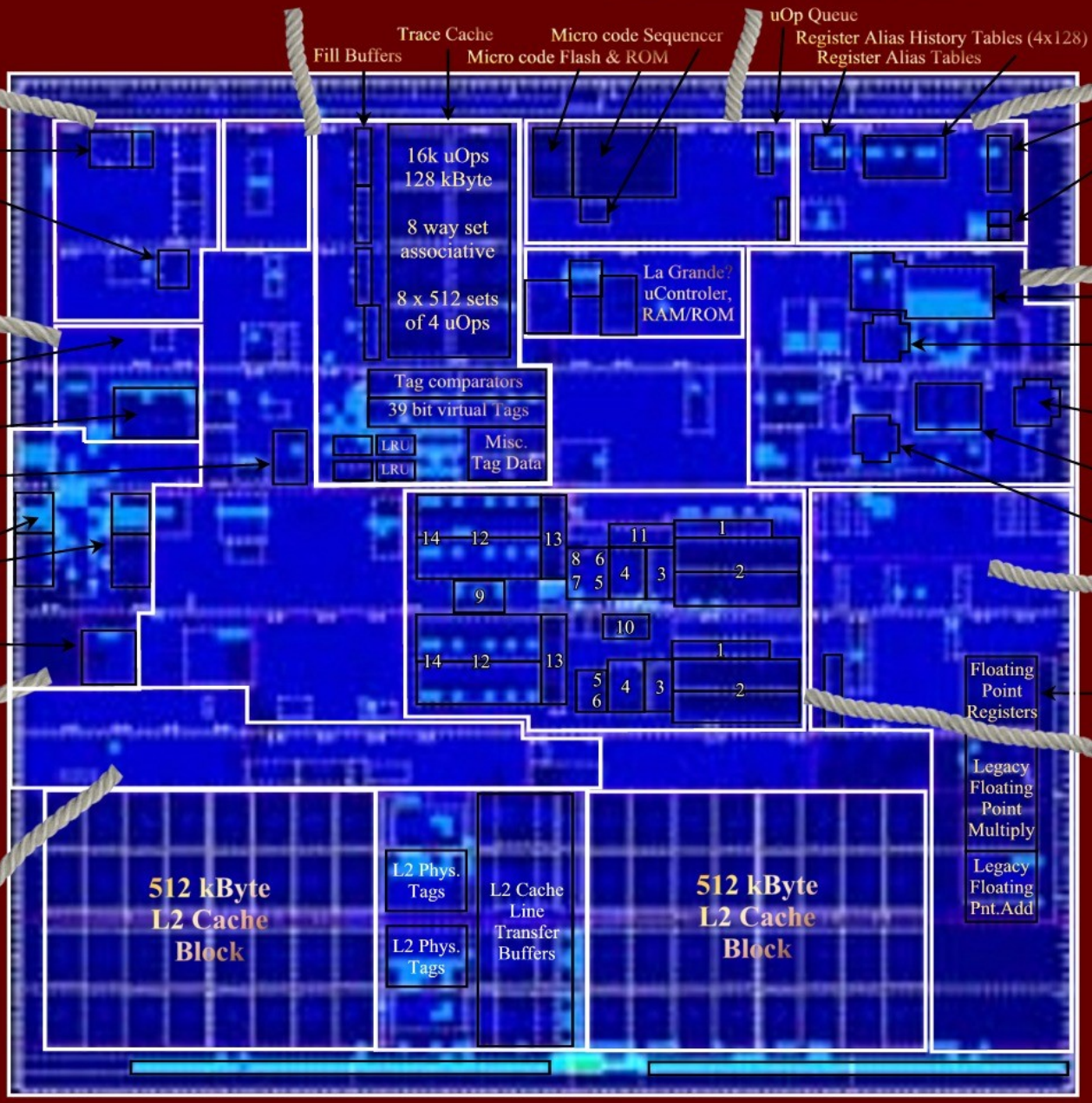
Instruction Fetch from L2 cache and Branch Prediction

Front Side Bus Interface, 533..800 MHz

Instruction Trace Cache

Execution Pipeline Start

Buffer Allocation & Register Rename



Instruction Queue (for less critical fields of the uOps)
General Instruction Address Queue & Memory Instruction Address Queue (queues register entries and latency fields of the uOps for scheduling)

uOp Schedulers

Parallel (Matrix) Scheduler for the two double pumped ALU's

General Floating Point and Slow Integer Scheduler: (8x8 dependency matrix)

FP Move Scheduler: (8x8 dependency matrix)

Load / Store Linear Address Collision History Table

Load / Store uOp Scheduler: (8x8 dependency matrix)

FP, MMX, SSE1..3

Floating Point, MMX, SSE1..3 Renamed Register File 256 entries of 128 bit.

Integer Execution Core

(1) uOp Dispatch unit & Replay Buffer Dispatches up to 6 uOps / cycle

(2) Integer Renamed Register File 256 entries of 32 bit (+ 6 status flags) 12 read ports and six write ports

(3) Databus switch & Bypasses to and from the Integer Register File.

(4) Flags, Write Back

(5) Double Pumped ALU 0

(6) Double Pumped ALU 1

(7) Load Address Generator Unit

(8) Store Address Generator Unit

(9) Load Buffer (96 entries)

(10) Store Buffer (48 entries)

(11) ROB Reorder Buffer 4x64 entries

(12) 16 kByte Level 1 Data cache four way set associative. 1R/1W

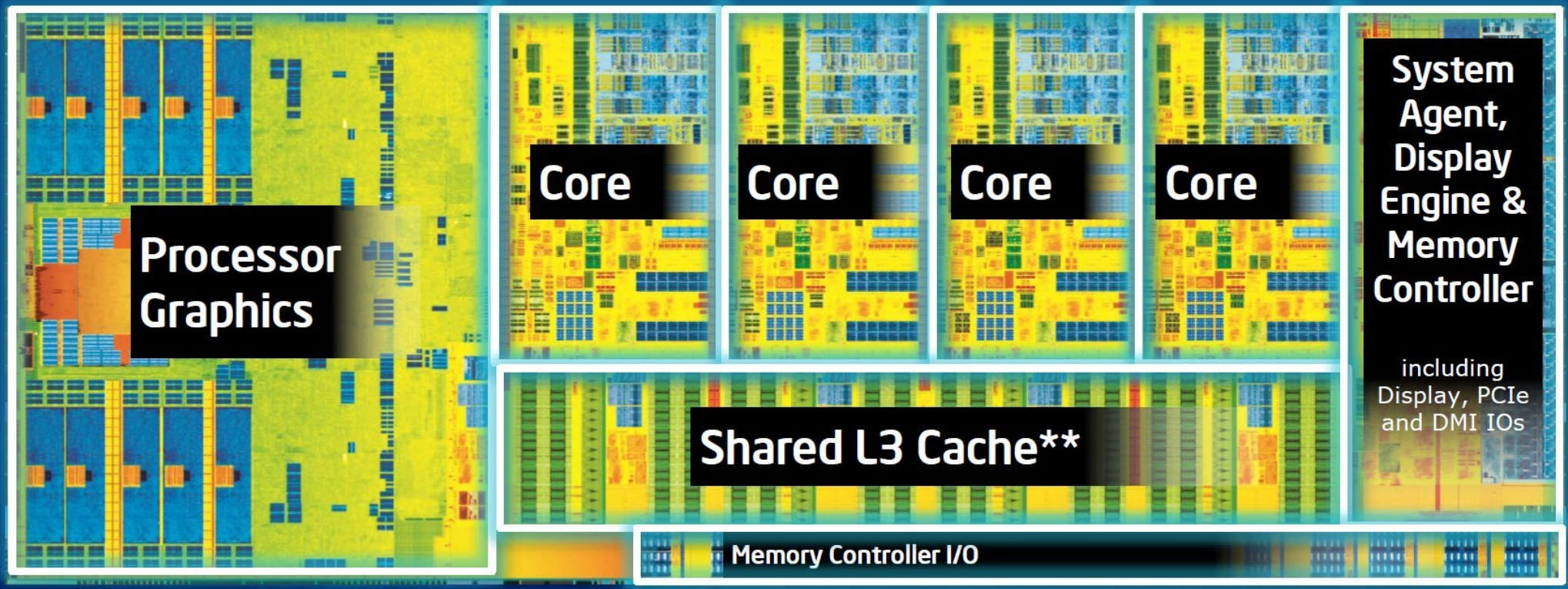
(13) Databus multiplexing

(14) Cache Line Read / Write Transferbuffers and 256 bit wide bus to and from L2 cache

April 19, 2003 www.chip-architect.com

4th Generation Intel® Core™ Processor Die Map

22nm Tri-Gate 3-D Transistors

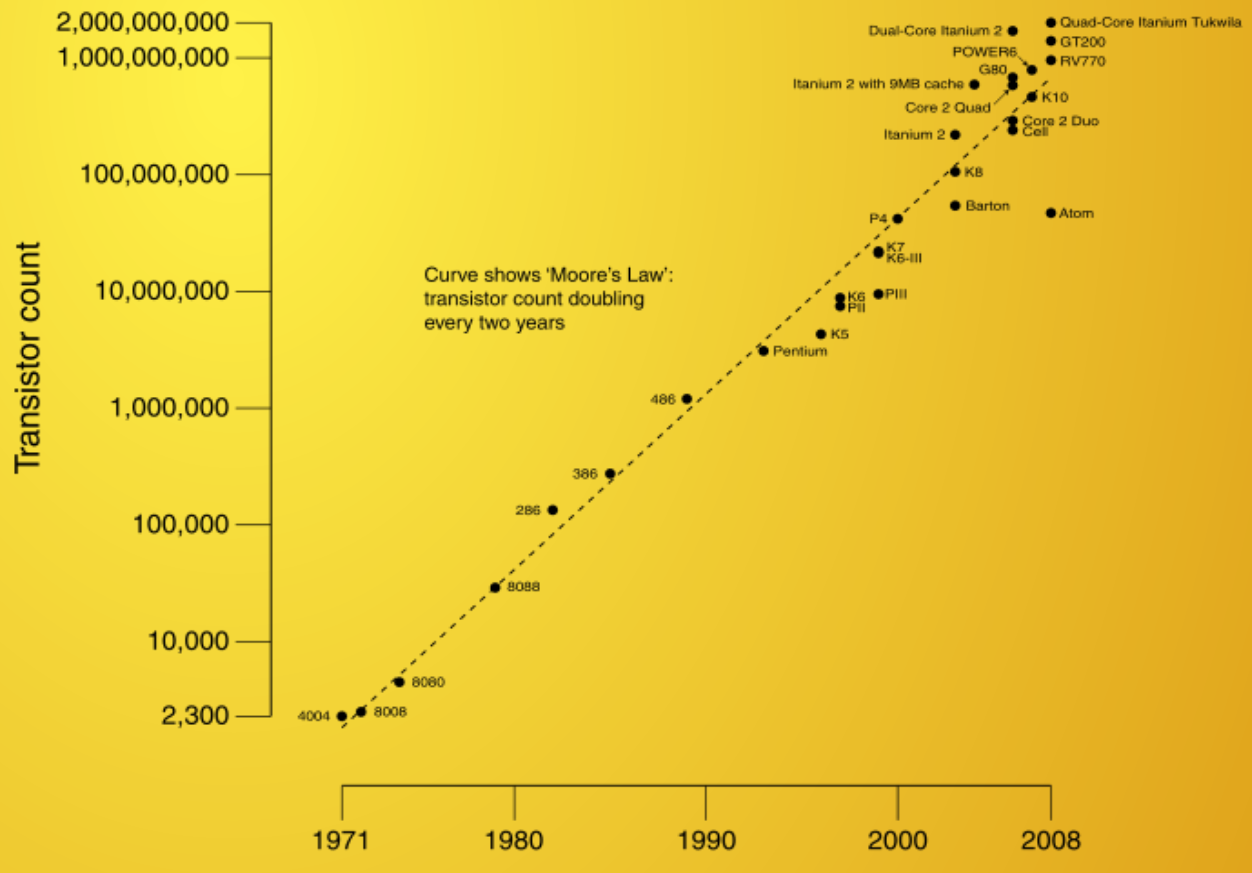


Quad core die shown above | Transistor count: 1.4 Billion | Die size: 177mm^2

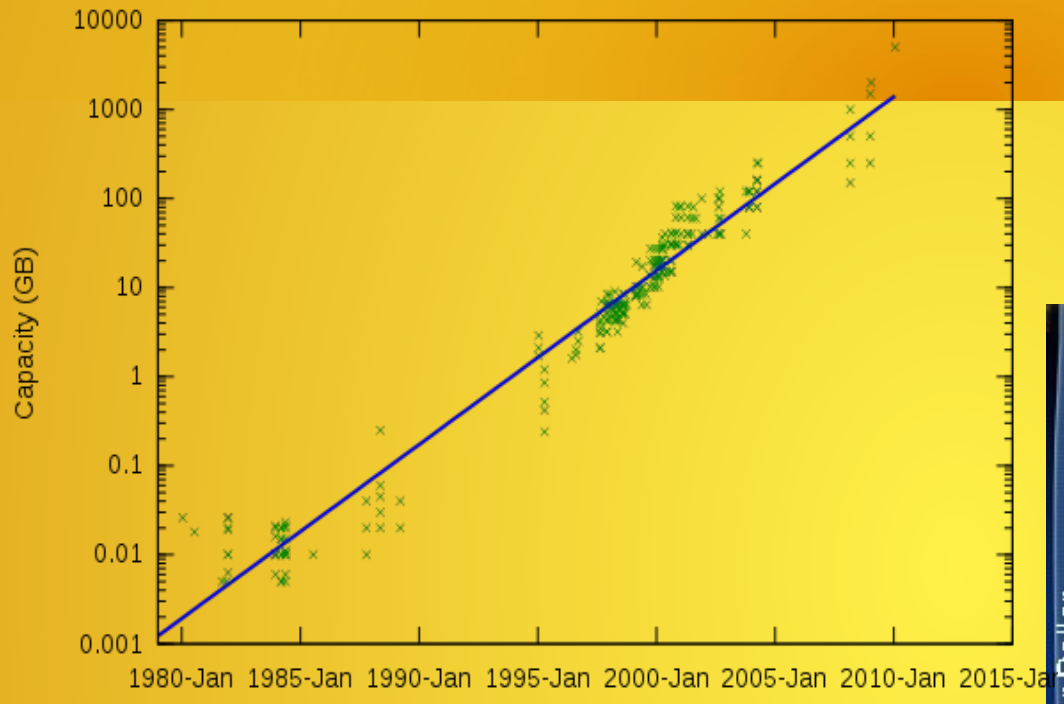
** Cache is shared across all 4 cores and processor graphics

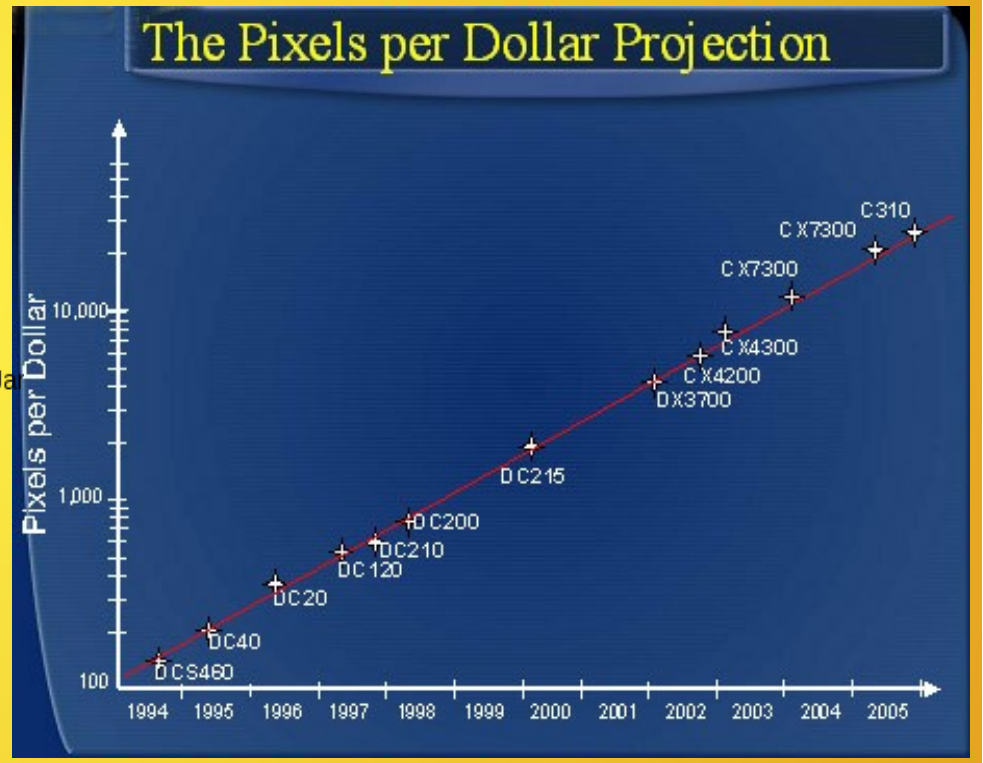
MOORE TÖRVÉNY

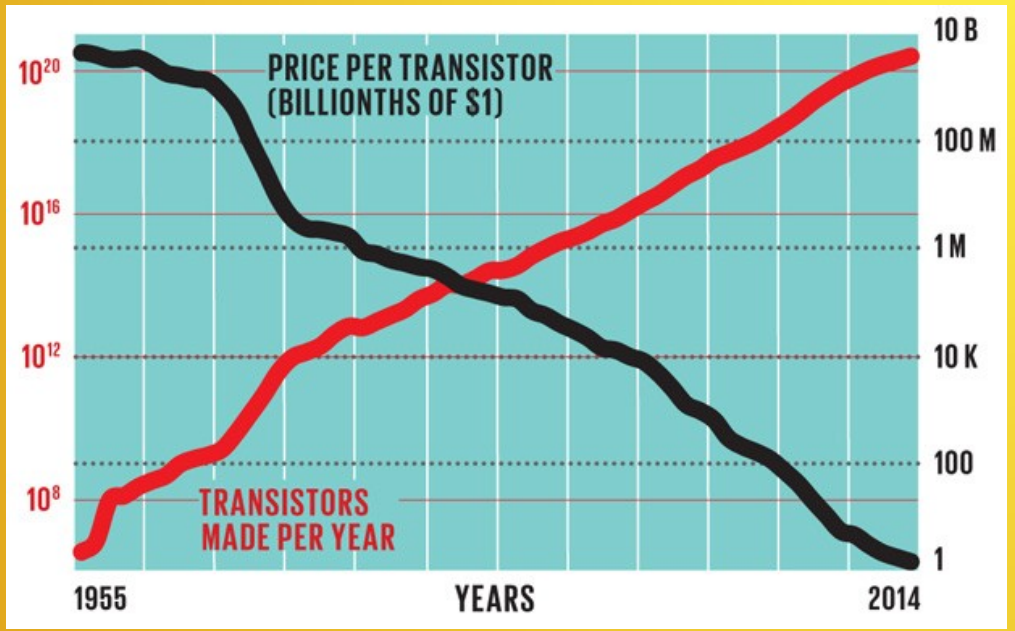
Gordon E. Moore, 1965: a processzorok sebessége közel megduplázódik 2 évente.
Más elektronikai komponensek is hasonlóan viselkednek!



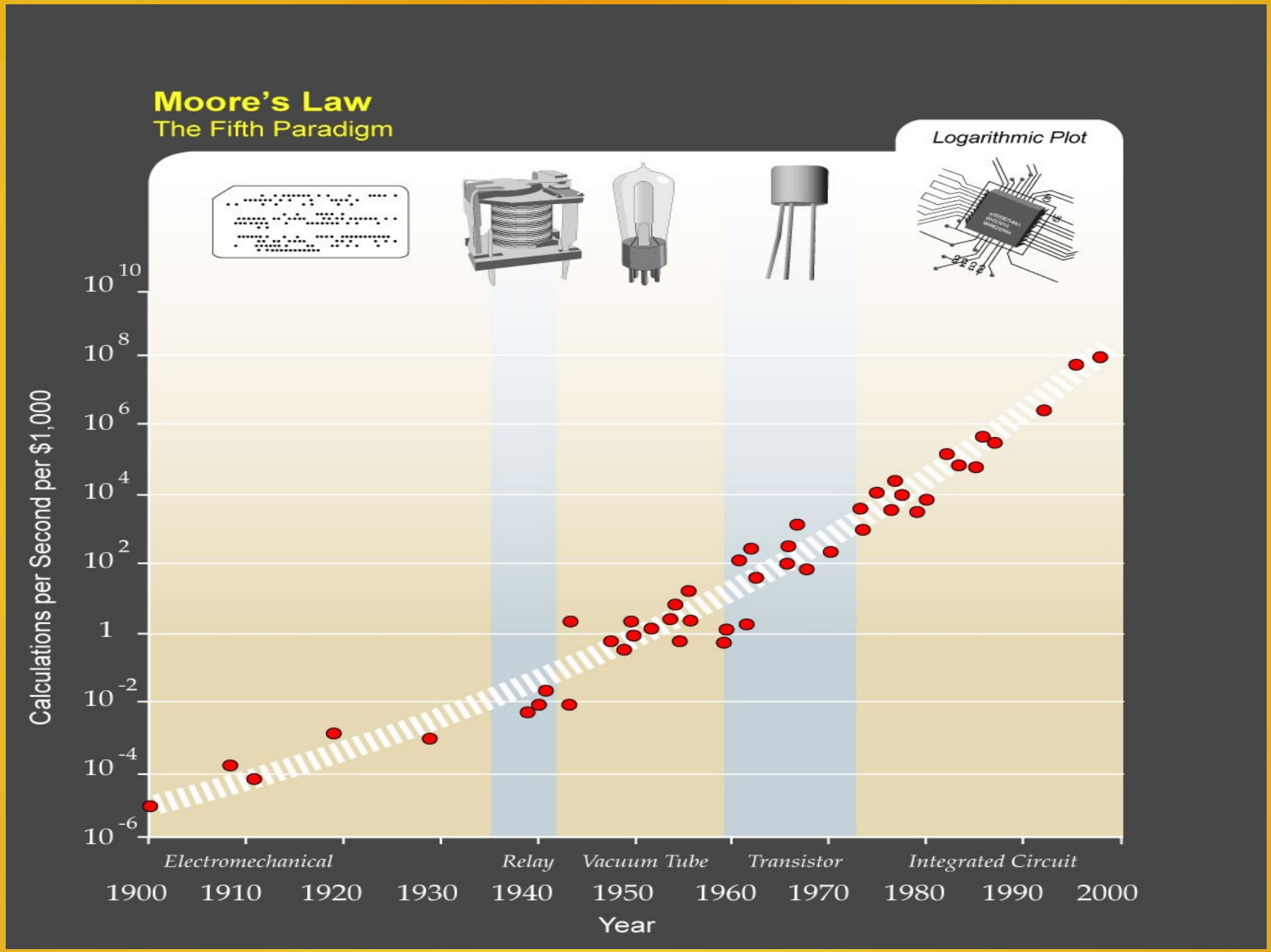
MOORE TÖRVÉNY



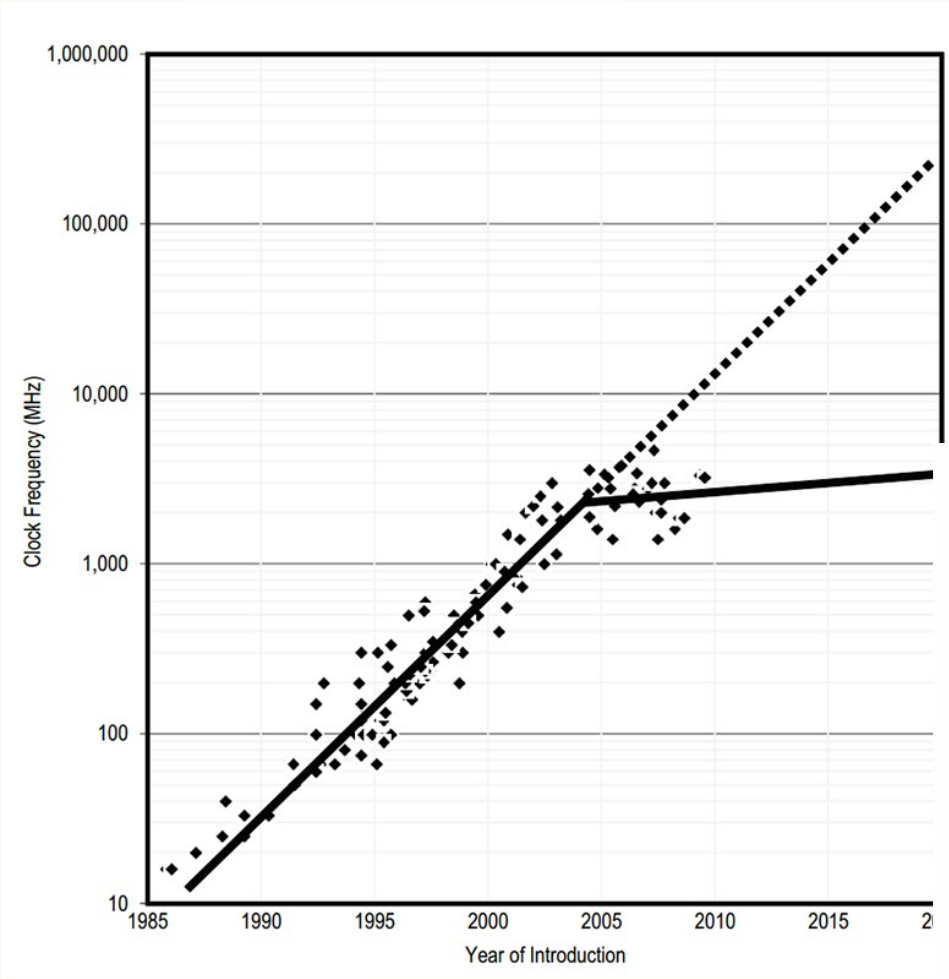


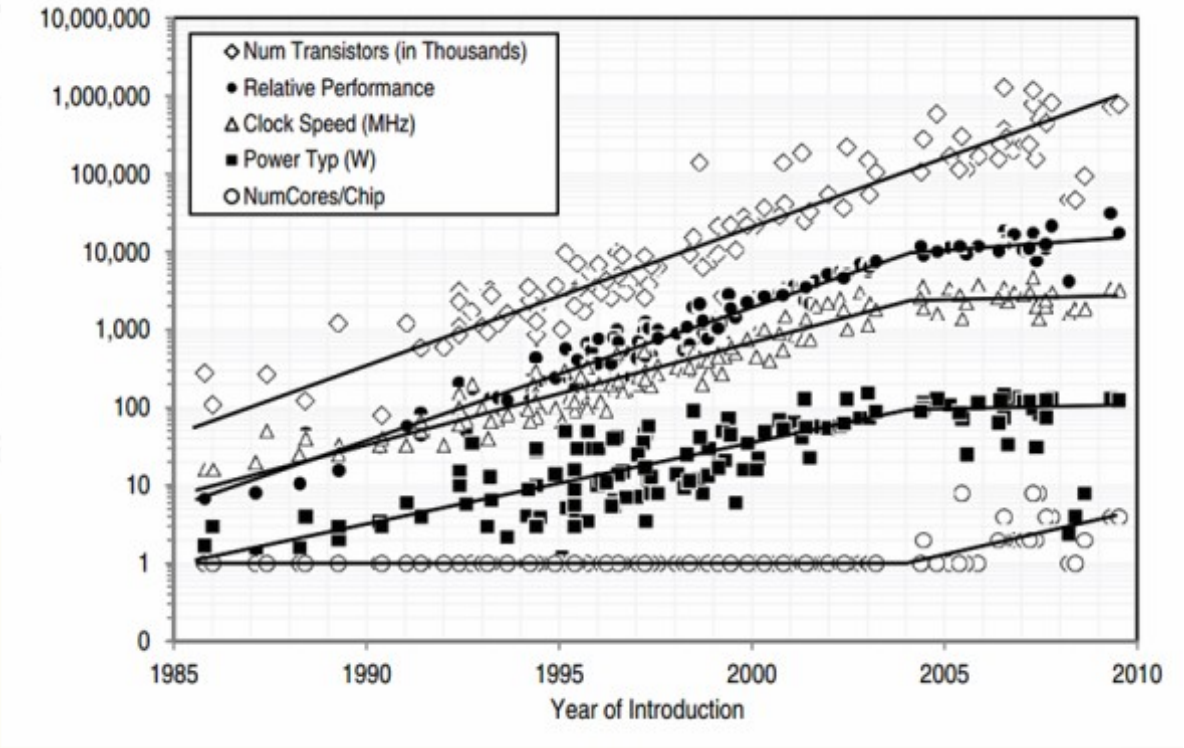


MOORE TÖRVÉNY

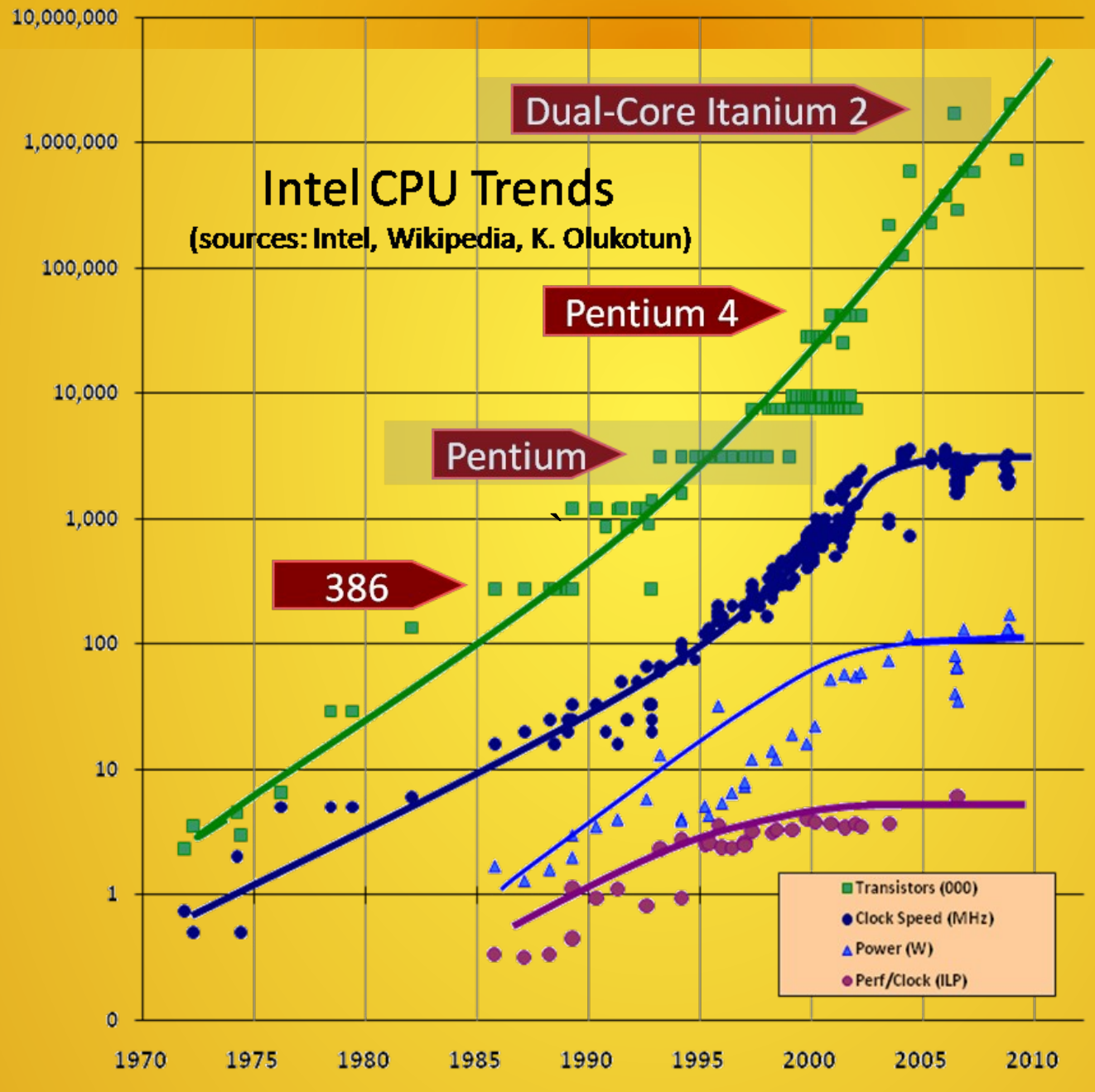


MOORE TÖRVÉNY





MOORE TÖRVÉNY



MIKROSZÁMÍTÓGÉPEK

Amint idővel tekintélyesebb (>kByte) mennyiségű memóriát is sikerült a processzor mellett egy chipen megvalósítani, úgy megszülettek az egy chipes MICROCOMPUTER-ek.

PIC
ATMEL/Arduino
Raspberry Pi

Ezek ma már évi milliárdos szériákban készülnek.