# CMP60001/COMP70086
# Advanced Computer Architecture
# Chapter 1.4

## The stored program concept and the Turing Tax

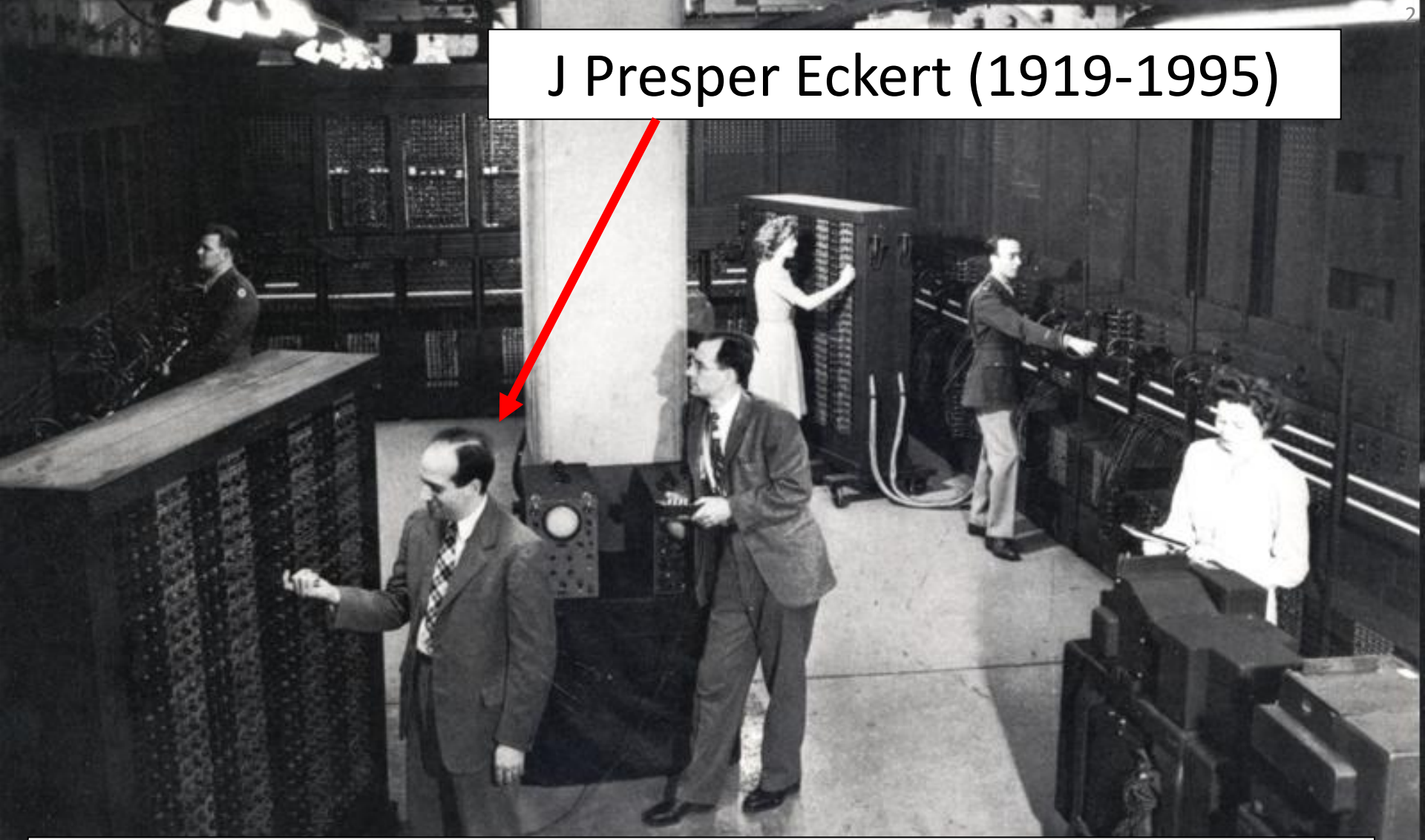
October 2025

Paul H J Kelly

These lecture notes are partly based on the course text, Hennessy and Patterson's *Computer Architecture, a quantitative approach (6th ed), and on the lecture slides of David Patterson's Berkeley course (CS252)*

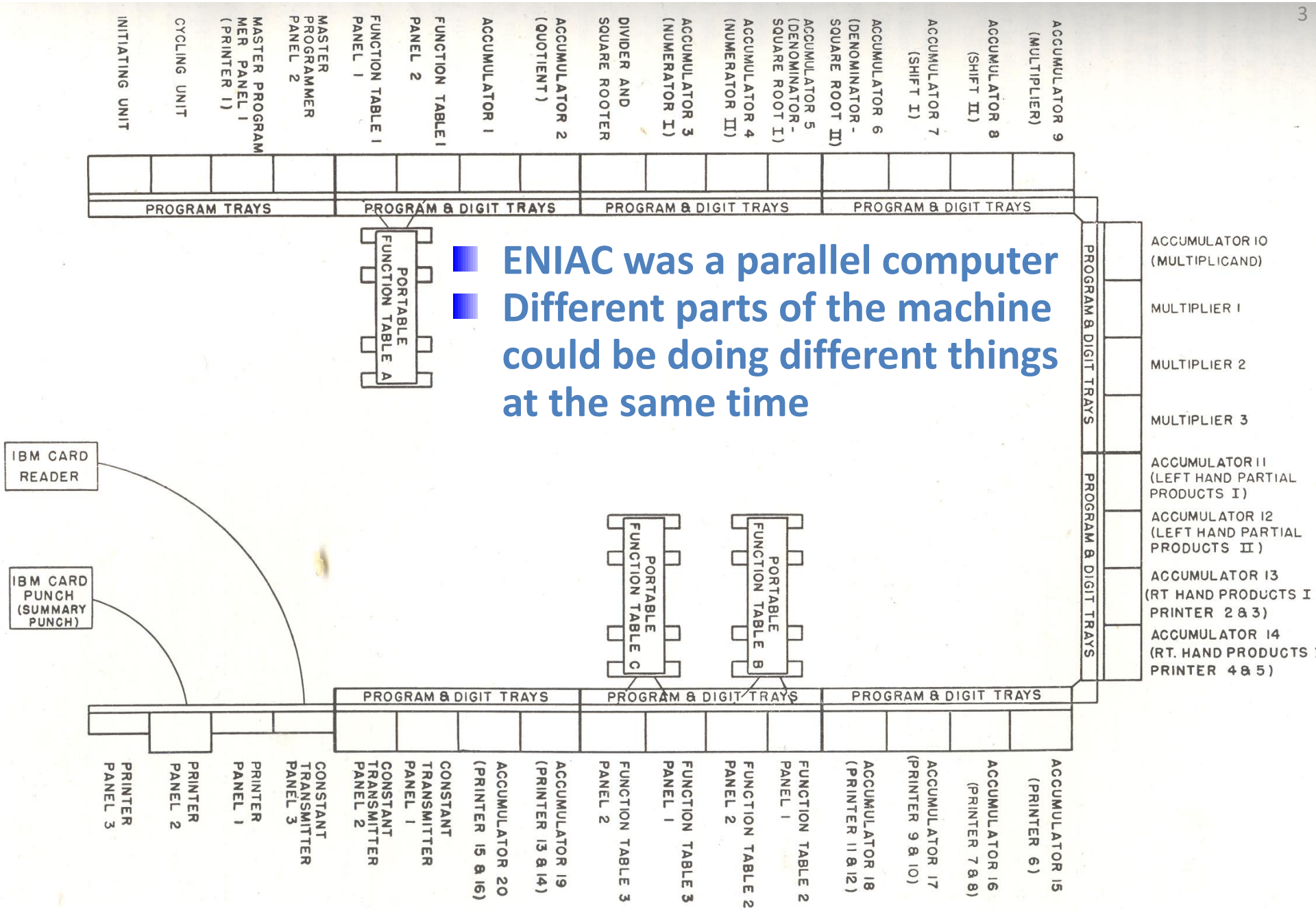
Course materials online on
https://scientia.doc.ic.ac.uk/2526/modules/60001/materials and
https://www.doc.ic.ac.uk/~phjk/AdvancedCompArchitecture/aca20/

# J Presper Eckert (1919-1995)



Co-inventor of, and chief engineer on, the ENIAC, arguably the first general-purpose computer (first operational Feb 14th 1946)

27 tonnes, 150KW, 5000 cycles/sec



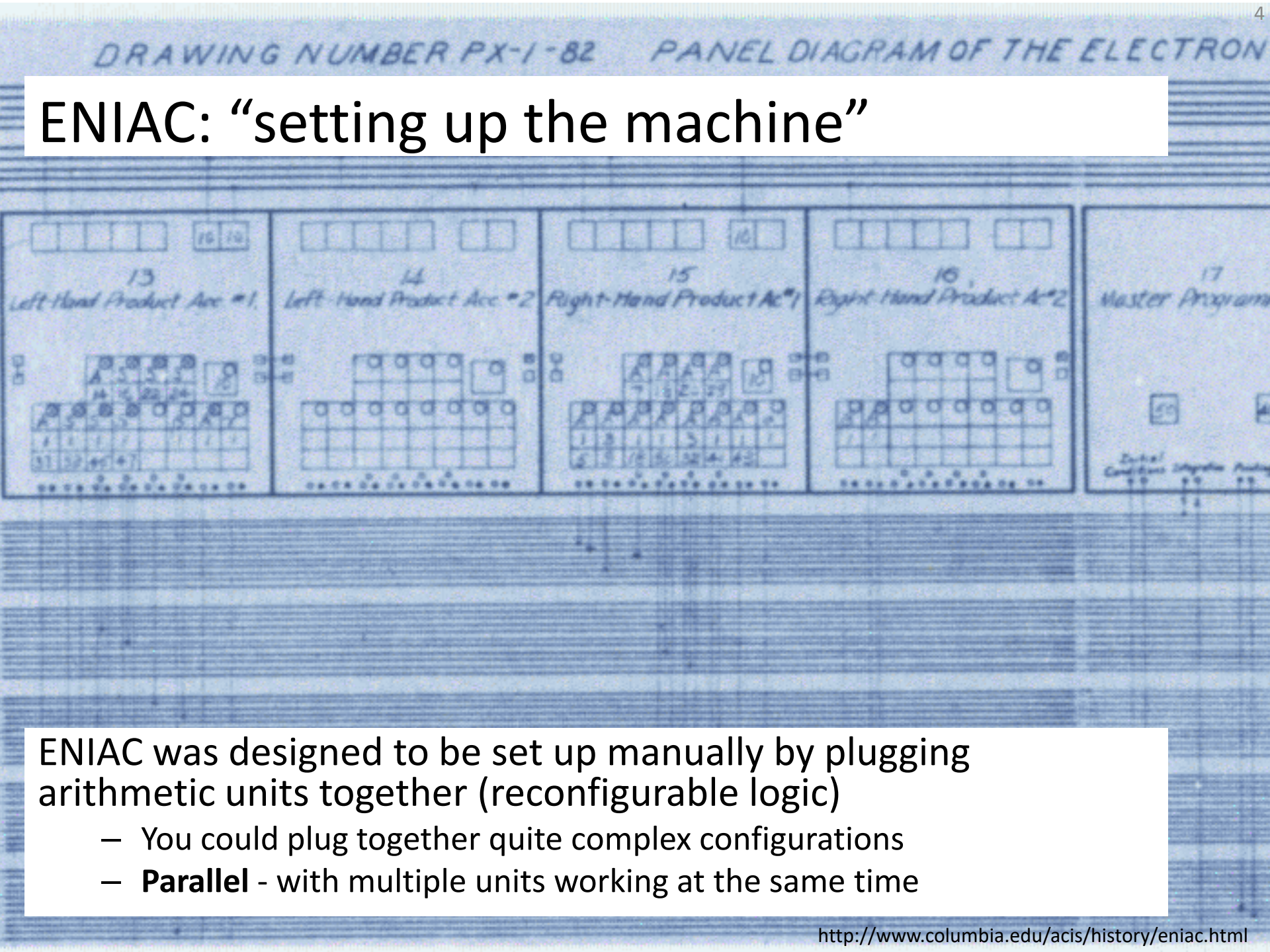

- ENIAC was a parallel computer
- Different parts of the machine could be doing different things at the same time

J.G. Brainerd & T.K. Sharpless. "The ENIAC." pp 163-172 Electrical Engineering, Feb 1948.

See also Eckert himself, https://www.youtube.com/watch?v=G8R6li54R20 ,

a google talk by Brian L Stuart on how it actually worked, https://www.youtube.com/watch?v=c-5n5J4wOig

and https://www.researchgate.net/profile/Edward_Davidson/publication/2985546_Introduction_to_The_ENIAC/links/56ec23b808aefd0fc1c7266f/Introduction-to-The-ENIAC.pdf

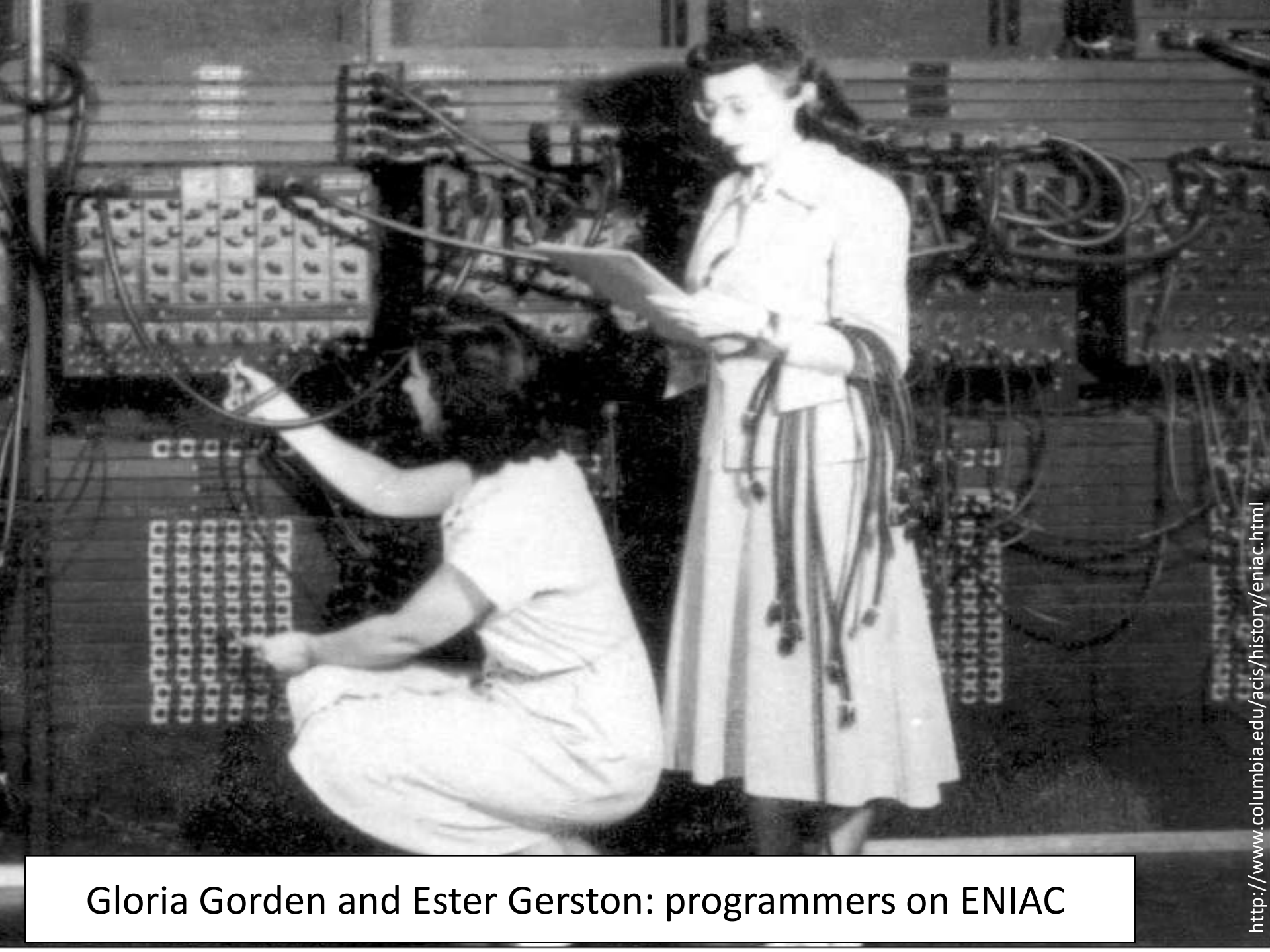



# ENIAC: "setting up the machine"

ENIAC was designed to be set up manually by plugging arithmetic units together (reconfigurable logic)

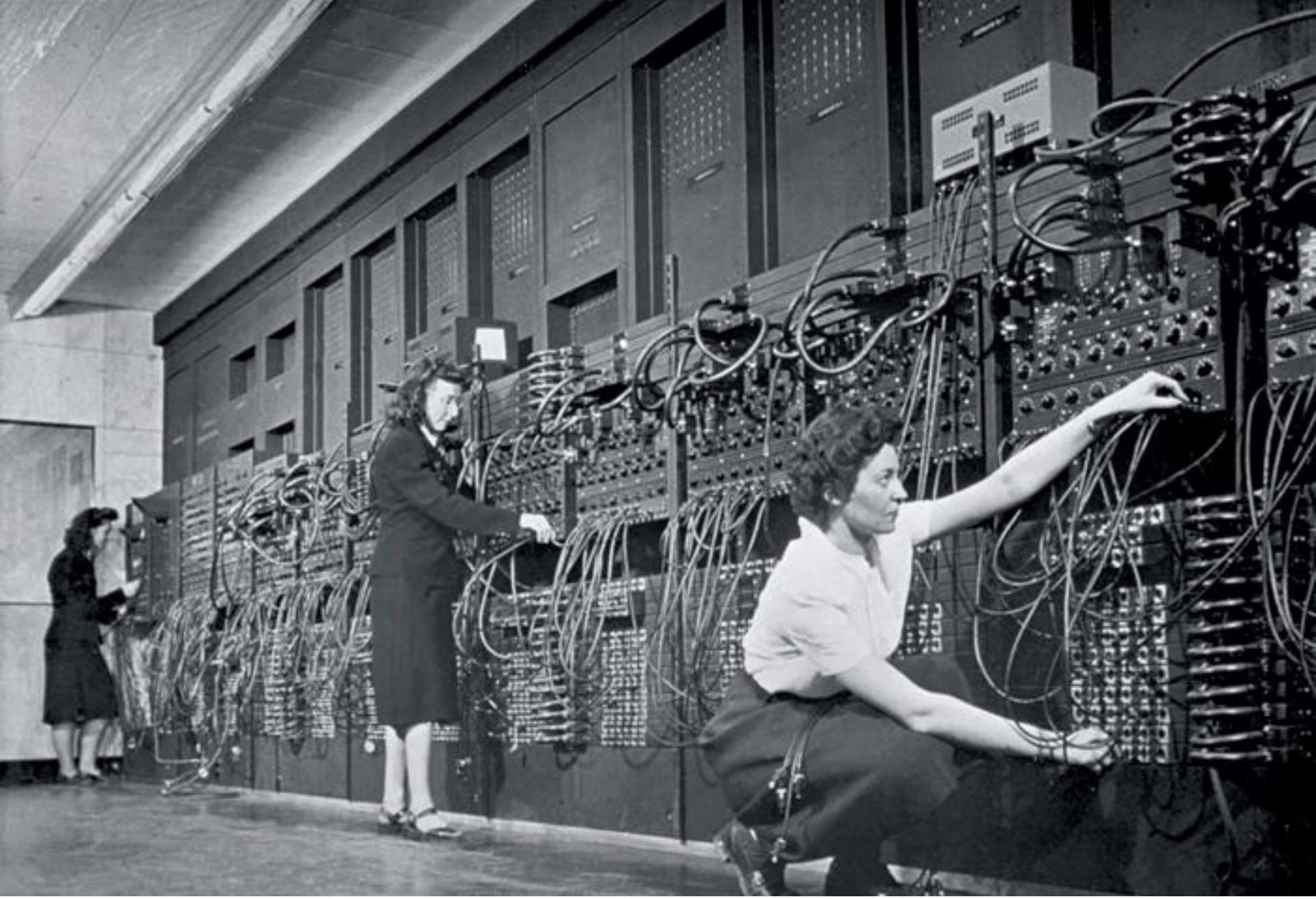
- You could plug together quite complex configurations
- **Parallel** - with multiple units working at the same time

http://www.columbia.edu/acis/history/eniac.html



Gloria Gorden and Ester Gerston: programmers on ENIAC

http://www.columbia.edu/acis/history/eniac.html



Jean Jennings (left), Marlyn Wescoff (center), and Ruth Lichterman program ENIAC

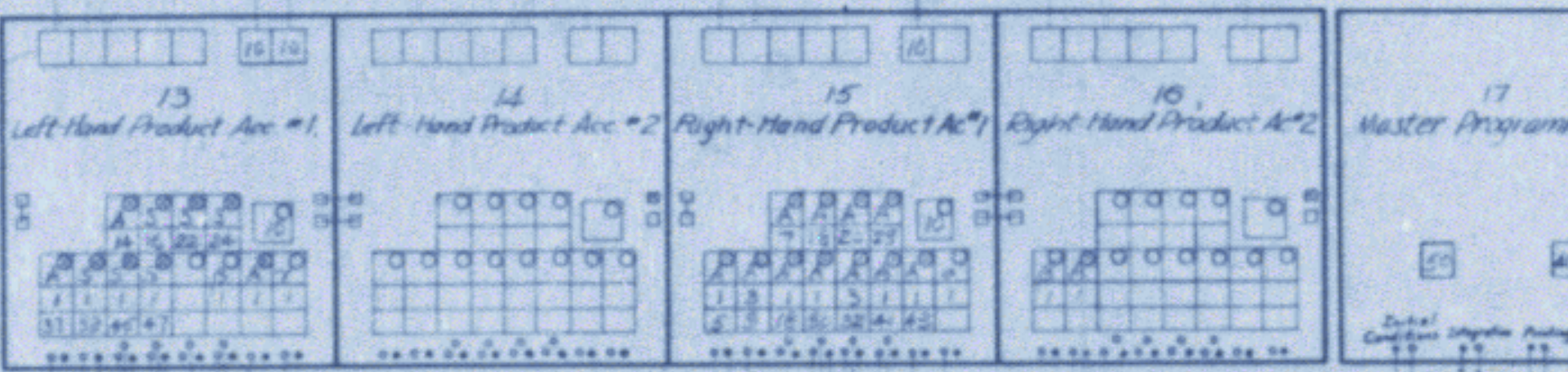
https://imgur.com/gallery/nh38c and http://fortune.com/2014/09/18/walter-isaacson-the-women-of-eniac/

# A PARALLEL CHANNEL COMPUTING MACHINE

Lecture by
J. P. Eckert, Jr.
Electronic Control Company

... Again I wish to reiterate the point that all the arguments for parallel operation are only valid provided one applies them to the steps which the built in or wired in programming of the machine operates. Any steps which are programmed by the operator, who sets up the machine, should be set up only in a serial fashion. It has been shown over and over again that any departure from this procedure results in a system which is much too complicated to use.

# ENIAC: “setting up the machine”




- The “big idea”: stored-program mode -
  - Plug the units together to build a machine that fetches instructions from memory - and executes them
  - So any calculation could be set up completely automatically – just choose the right sequence of instructions

http://www.columbia.edu/acis/history/eniac.html

We now formulate a set of instructions to effect this 4-way decision between (α)–(δ). We state again the contents of the short tanks already assigned:

$\overline{1}_1$) $\mathcal{N} n'_{(-30)}$ $\overline{2}_1$) $\mathcal{N} m'_{(-30)}$ $\overline{3}_1$) $\mathcal{N} x^{\circ}_{m'}$ $\overline{4}_1$) $\mathcal{N} y^{\circ}_{m'}$

$\overline{5}_1$) $\mathcal{N} n_{(-30)}$ $\overline{6}_1$) $\mathcal{N} m_{(-30)}$ $\overline{7}_1$) $\mathcal{N} 1_{\alpha(-30)}$ $\overline{8}_1$) $\mathcal{N} 1_{\beta(-30)}$

$\overline{9}_1$) $\mathcal{N} 1_{\delta(-30)}$ $\overline{10}_1$) $\mathcal{N} 1_{\delta(-30)}$ $\overline{11}_1$) ... → $\mathcal{C}$

Now let the instructions occupy the (long tank) words $1_1, 2_1, \ldots$ :

$1_1$) $T_1 - \overline{5}_1$

$2_1$) $\overline{9}_1$ s $\overline{7}_1$

$3_1$) $\sigma \rightarrow \overline{12}_1$

$4_1$) $\overline{1}_1 - \overline{5}_1$

$5_1$) $\overline{10}_1$ s $\overline{8}_1$

$6_1$) $\sigma \rightarrow \overline{12}_1$

$7_1$) $\overline{2}_1 - \overline{6}_1$

$8_1$) $\overline{13}_1$ s $\overline{12}_1$

$9_1$) $\sigma \rightarrow \overline{11}_1$

$\overline{11}_1$) $1_\alpha 1_\beta 1_\gamma 1_\delta \rightarrow \mathcal{C}$

$\sigma$) $\mathcal{N} n' - n_{(-30)}$

$\sigma$) $\mathcal{N} {}^{1\delta}_{1\alpha}{}_{(-30)}$ for $n' \leq n$

$\overline{12}_1$) $\mathcal{N} {}^{1\delta}_{1\alpha}{}_{(-30)}$ for $n' \geq n$

$\sigma$) $\mathcal{N} n' - n_{(-30)}$

$\sigma$) $\mathcal{N} {}^{1\delta}_{1\beta}{}_{(-30)}$ for $n' \leq n$

$\overline{13}_1$) $\mathcal{N} {}^{1\delta}_{1\beta}{}_{(-30)}$ for $n' \geq n$

$\sigma$) $\mathcal{N} m' - m_{(-30)}$

$\sigma$) $\mathcal{N} {}^{-\overline{4}_1)\ldots}_{-\overline{11}_1)\ldots}$ for $m' \leq m$

i.e.

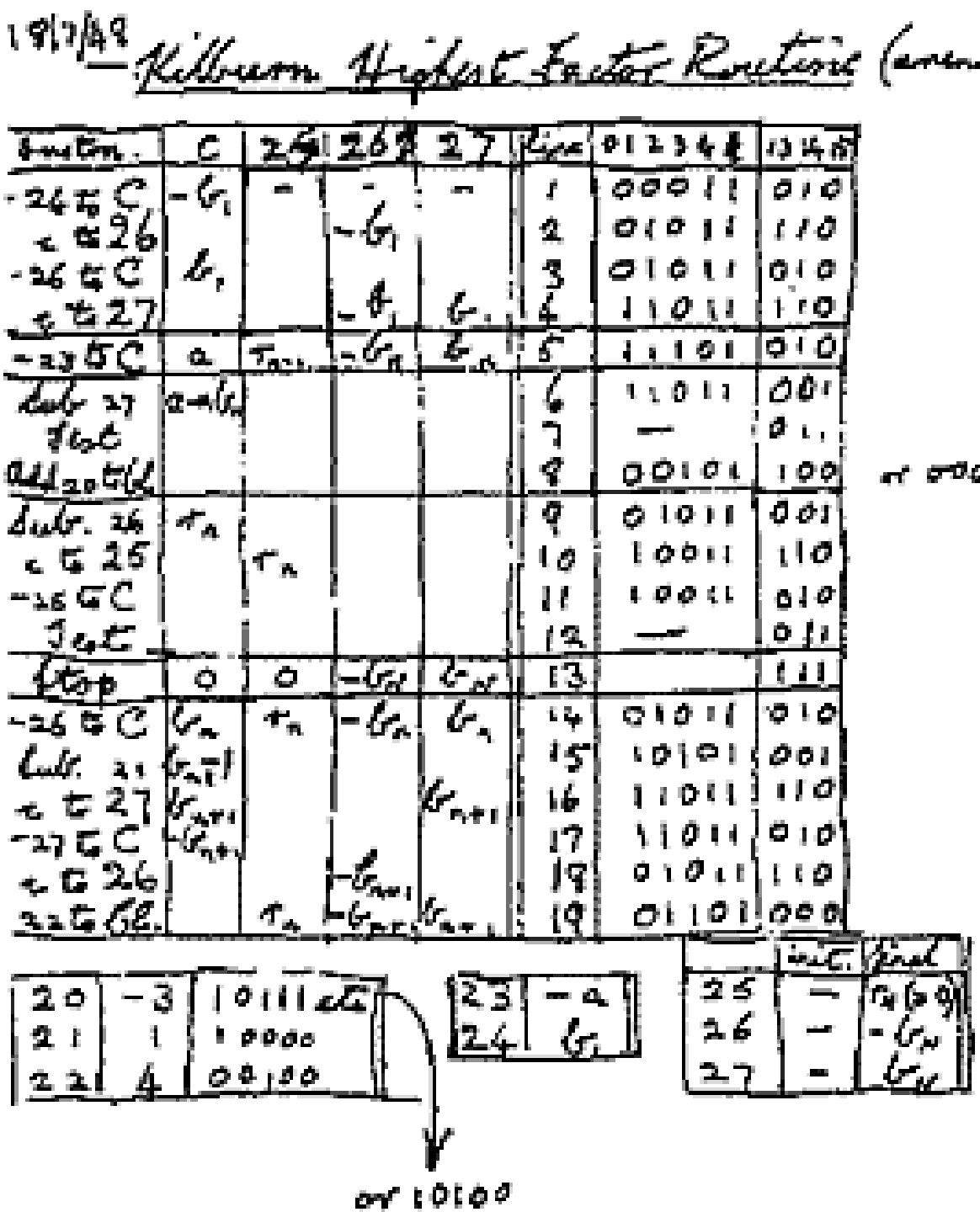
$\sigma$) $\mathcal{N} {}^{1\delta\ 1\beta}_{1\gamma\ 1\alpha}{}_{(-30)}$ for $m' > m, n' > n$ $m' > m, n' < n$ $m' < m, n' > n$ $m' < m, n' < n$

i.e for $(\delta)(\beta)$ $(\gamma)(\alpha)$, respectively.

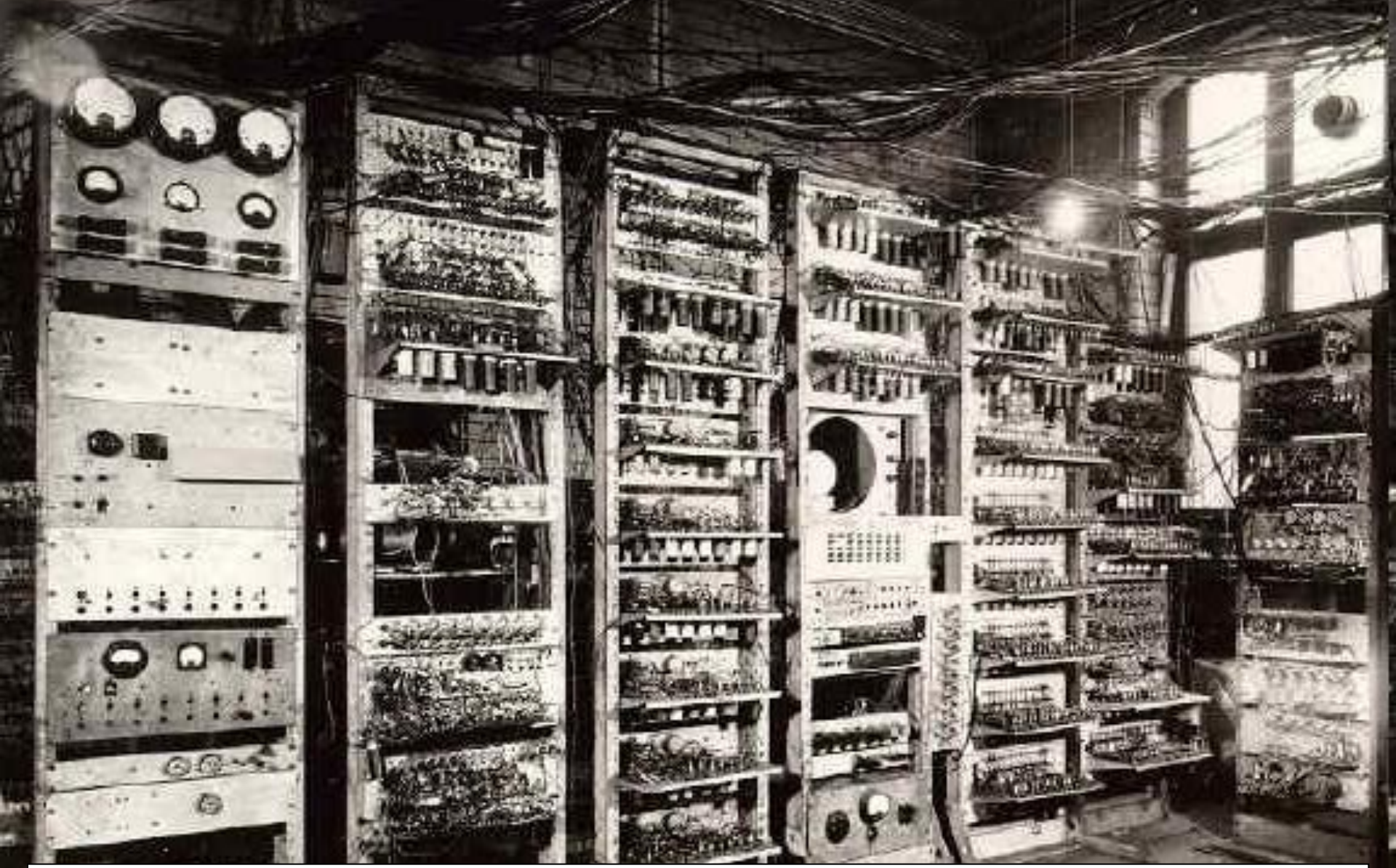
for $(\alpha), (\beta), (\gamma), (\delta)$, respectively.

- John von Neumann wrote his first "program" in 1945
- It's clear he had the stored program idea in mind
- It was a couple of years before a machine to do it was actually built


- Knuth, D. E. 1970. Von Neumann's First Computer Program. ACM Comput. Surv. 2, 4 (Dec. 1970), 247-260.

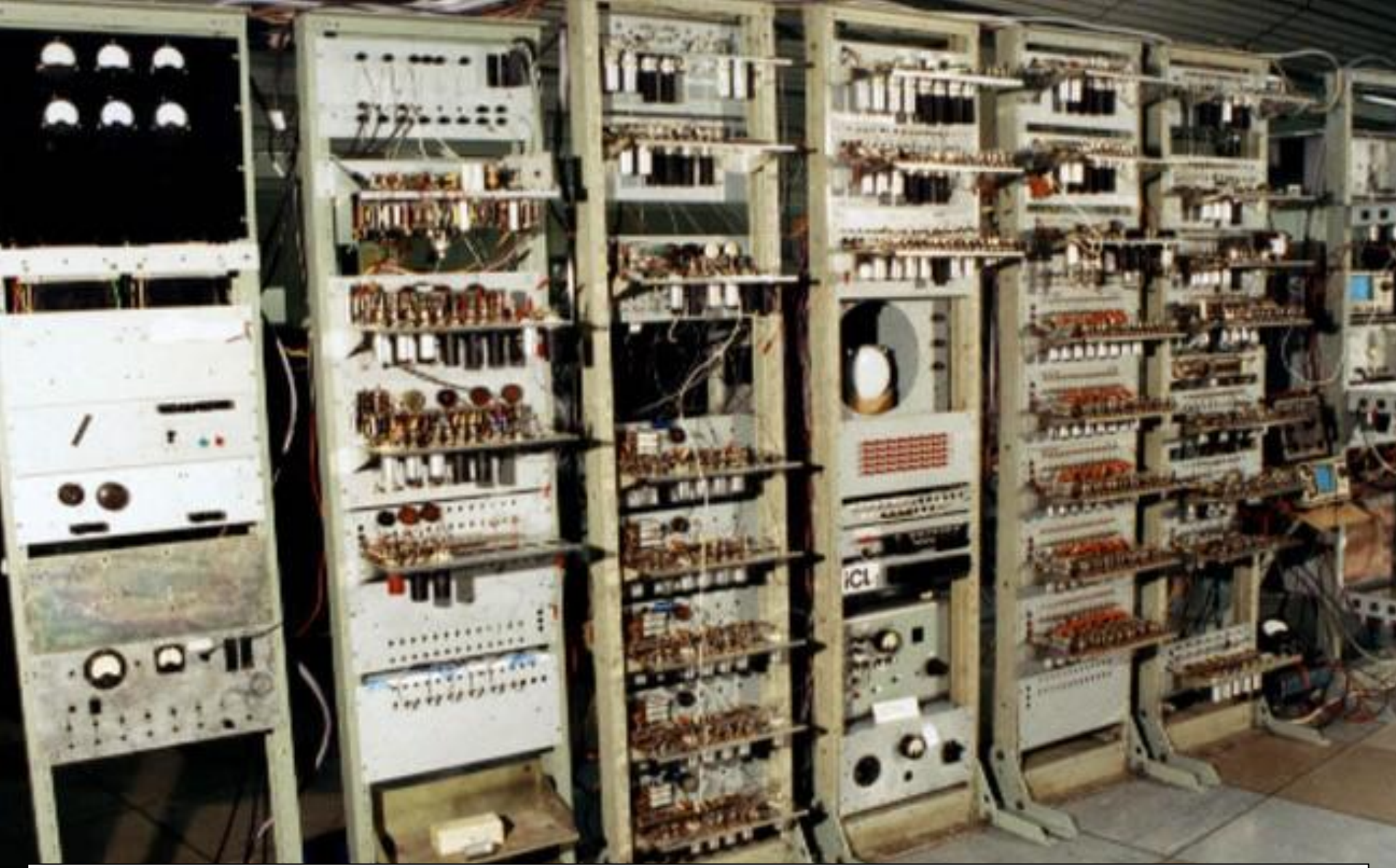



- This is the first program to actually run!



Manchester Small-Scale Experimental Machine (SSEM), nicknamed Baby

Ran its first program on 21 June 1948 – the first program ever!

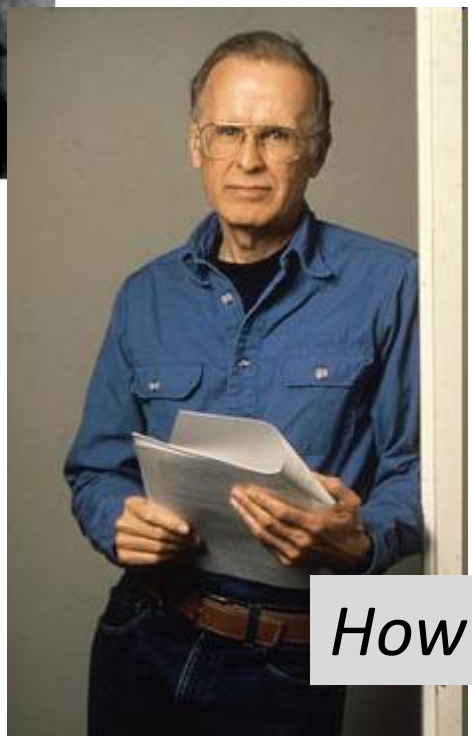



Manchester Small-Scale Experimental Machine (SSEM), nicknamed Baby

Rebuilt for the 60th anniversary, now in in the Museum of Science and Industry in Manchester

# The "von Neumann bottleneck"


John von Neumann
http://en.wikipedia.org/wiki/John_von_Neumann



John Backus
"**Can Programming be Liberated from the von Neumann Style?"** (1979)

www.post-gazette.com/pg/07080/771123-96.stm

The price to pay:

- **Stored-program mode was serial – one instruction at a time**

- How can we have our cake - and eat it?
  - **Flexibility and ease of programming**
  - **Performance of parallelism**

*How to beat the "Turing Tax"*

# Alan Turing

FOLLOW

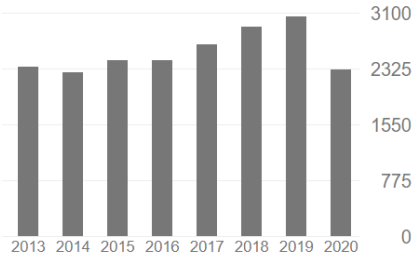
Reader, University of Manchester
Verified email at lsbu.ac.uk - Homepage

Mathematics Computer Science Cryptography Artificial Intelligence Morphogenesis

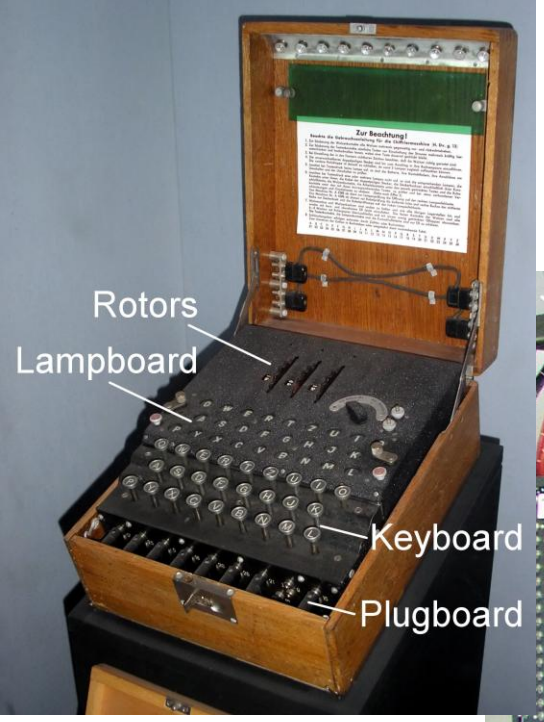
| TITLE | CITED BY | YEAR |
|---|---|---|
| Computing machinery and intelligence<br>AM Turing<br>Computers & Thought, 11-35 | 14382 * | 1995 |
| The imitation game<br>AM Turing<br>Theories of Mind: An introductory reader, 51 | ~~14287~~ * | 2006 |
| The chemical basis of morphogenesis<br>AM Turing<br>Bulletin of Mathematical Biology 52 (1), 153-197 | 13458 * | 1952 |
| The chemical basis of morphogenesis<br>AM Turing<br>Bulletin of Mathematical Biology 52 (1-2), 153-197 | ~~13372~~ | 1990 |
| On computable numbers, with an application to the Entscheidungsproblem: A correction<br>AM Turing<br>Proceedings of the London Mathematical Society 43 (2), 544-546 | 11923 * | 1937 |
| On computable numbers, with an application to the Entscheidungsproblem<br>AM Turing<br>Proceedings of the London Mathematical Society 42 (2), 230-265 | ~~11767~~ | 1936 |
| Systems of logic based on ordinals<br>AM Turing<br>Proceedings of the London Mathematical Society, Series 2 45, 161-228 | 1017 | 1939 |
| Intelligent machinery<br>AM Turing<br>The Essential Turing, 395-432 | 897 * | 1948 |
| Intelligent machinery, a heretical theory (c. 1951)<br>AM Turing<br>The Essential Turing, 465-475 | ~~894~~ * | 2004 |
| Rounding-off errors in matrix processes<br>AM Turing<br>The Quarterly Journal of Mechanics and Applied Mathematics 1 (1), 287-308 | 521 | 1948 |
| Computability and λ-definability<br>AM Turing<br>The Journal of Symbolic Logic 2 (4), 153-163 | 429 | 1937 |
| Checking a large routine<br>AM Turing<br>The early British computer conferences, 70-72 | 418 * | 1948 |

Cited by VIEW ALL

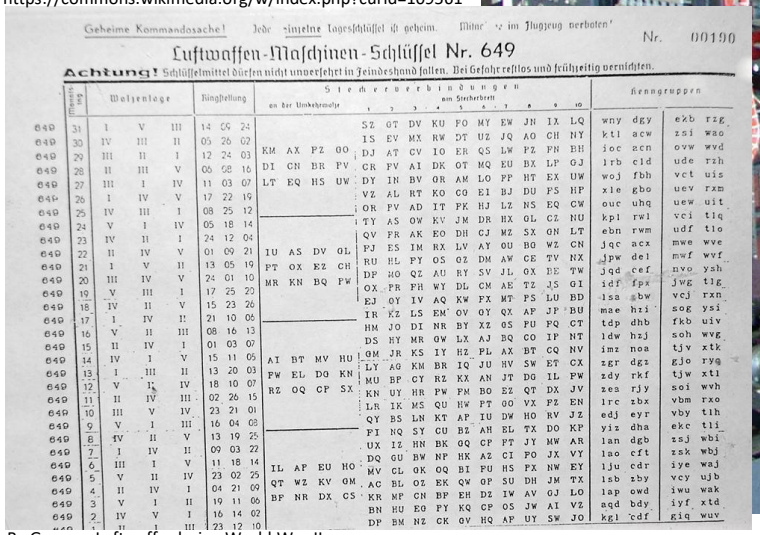
| | All | Since 2015 |
|---|---|---|
| Citations | 48417 | 15865 |
| h-index | 42 | 24 |
| i10-index | 99 | 54 |



# Alan Turing worked on a couple of projects in his career
# One of them was defeating Nazi Germany in WW2




By Karsten Sperling, http://spiff.de/photo - Own work - Derivative of author/uploader's own work -This file was derived from: EnigmaMachine.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=109561



Geheime Kommandosache! Jede einzelne Tagesschlüssel ist geheim. Mitnahme im Flugzeug verboten!

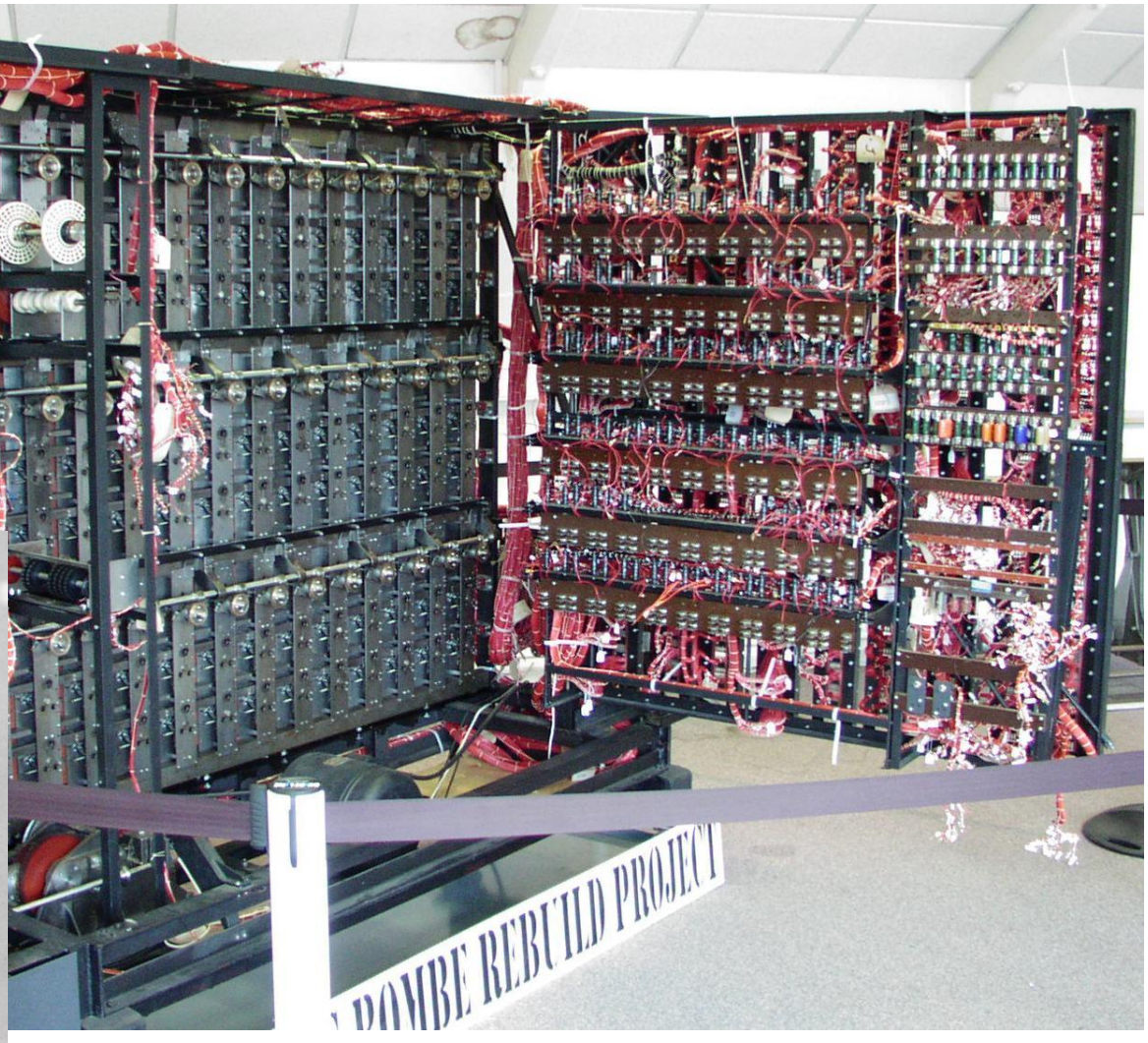
Nr. 00190

**Luftwaffen-Maschinen-Schlüssel Nr. 649**

**Achtung!** Schlüsselmittel dürfen nicht unversehrt in Feindeshand fallen. Bei Gefahr restlos und frühzeitig vernichten.

| Monats-tag | Walzenlage | Ringstellung | Steckerverbindungen am Steckerbrett an der Umkehrwalze | Kenngruppen |
|---|---|---|---|---|
| 30 | I V III | II 05 26 02 | SZ GT DV KU FO MY EW JN IX LQ / IS EV MX RW DT UZ JQ AO CH NY / DJ AT CV IO ER QS LW PZ FN BH / CR PV AI DK OT MQ EU BX LP GJ / DY IN BV OR AM LO FP HT EX UW / VZ AL HT KO CO EI BJ DU FS HP / QR PV AD IT PK HJ LZ NS EQ CW / TY AS OW KV JM DR HX GL CZ NU / QV FR AK EO DH CJ MZ SX ON LT / PJ ES IM RX LV AY OU BQ WZ CN / RU HL FY OS OZ DM AW CE TV NX / DP HO QZ AU RY SV JL QX BE TW / OX PR FH WY DL CM AE TZ JS GI / EJ OY IV AQ KW FX MT PS LU BD / IR KZ LS EM OV OY QX AF JP BU / HM JO DI NR BY XZ GS FU FQ CT / DS HY MR GW LX AJ BQ CO IP NT / GM JR KS IY HZ PL AX BT CQ NV / LY AG KM BR IQ JU HV SW ET CX / MU BP GY RZ KX AN JT DG IL FW / KN UY HR PW FM BO EZ QT DX JV / LR IK MS QU HW PT GO VX PZ EN / QY BS LN KT AP IU DW HO RV JZ / FI NQ SY GU BZ AH EL TX DO KP / UX IZ HN BK QQ GP FT JY MW AR / DQ GU BW NP HK AZ CI FO JX VY / MV GL QK OQ BI FU HS FX NW EY / AG BL OZ EK QW GP SU DH JM TX / KR MP CN BP EH DZ IW AV GJ LO / BN HU EO FY KQ CP OS JW AI VZ / DP BM NZ CK QV HQ AP UY SW JO | wny dgy ezb rzg / ktl acw zsi wao / joc zcn ovw wvd / lrb cld ude rzh / woj fbh vct uis / xle gbo uev rxm / ouc uhq uew uit / kpl rwl vci tlq / ebn rwm udf tlo / jqc acx mwe wve / jpw del mwf wvf / jqd cef nvo ysh / idf fpx jwg tlg / lsa sbw vcj rxn / mae hri sog ysi / tdp dhb fkb uiv / ldw hzj soh wvg / imz noa tjv xtk / zgr dgz gjo ryg / zdy rkf tjw xtl / zea rly soi wvh / lrc zbx vbm rxo / edj eyr vby tih / yiz dha ekc tli / lan dgb zsj wbi / lao cft zsk wbj / lju cdr iye waj / lsb zby vcy ujb / lap owd iwu wak / aqd bdy iyf xtd / kgl cdf giq wuv |
| 1 | V II IV | I 23 12 10 | | |

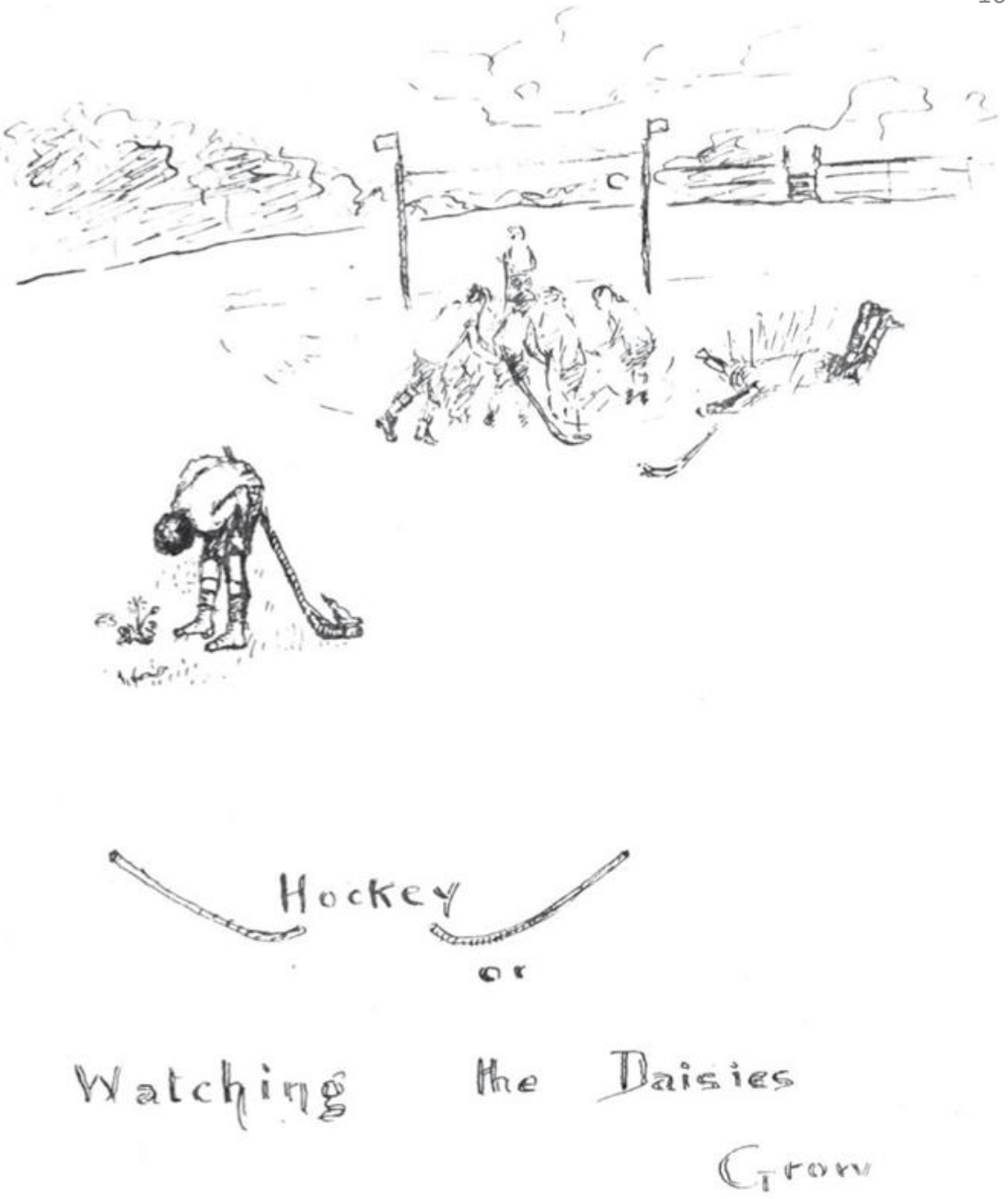
By German Luftwaffe during World War II - http://jproc.ca/crypto/enigma_keylist_3rotor_b.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=44261334




A complete and working replica of a bombe now at The National Museum of Computing on Bletchley Park, https://en.wikipedia.org/wiki/Alan_Turing

# The “Turing Tax”

Discussion exercise




https://www.commondreams.org/sites/commondreams.org/files/imce-images/turing_mother_drawing.jpg

# ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

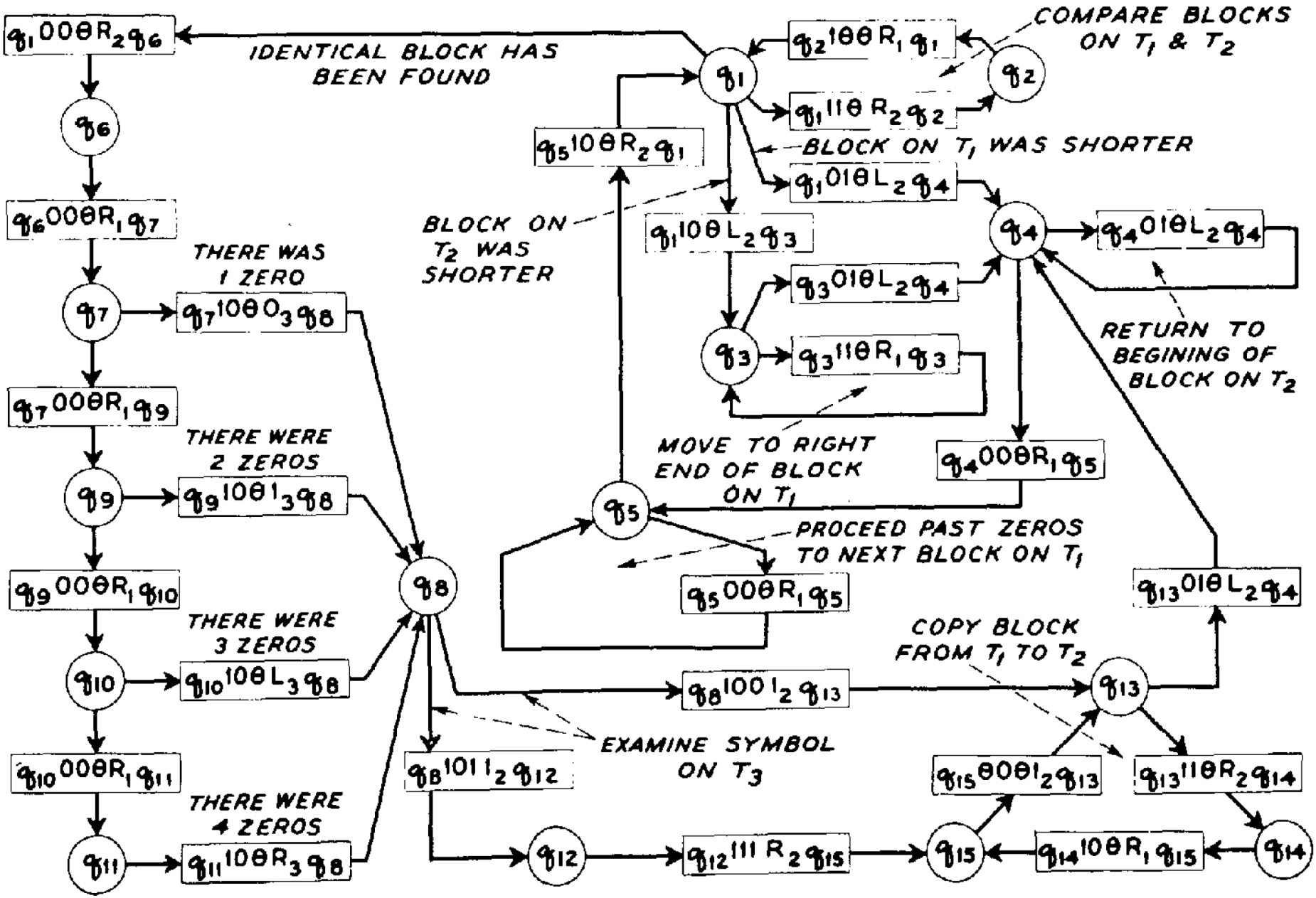
*By* A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

## 6. *The universal computing machine.*

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathfrak{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathfrak{M}$, then $\mathfrak{U}$ will compute the same sequence as $\mathfrak{M}$. In this section I explain in outline the behaviour of the machine. The next section is devoted to giving the complete table for $\mathfrak{U}$.

# 15-STATE UNIVERSAL TURING MACHINE

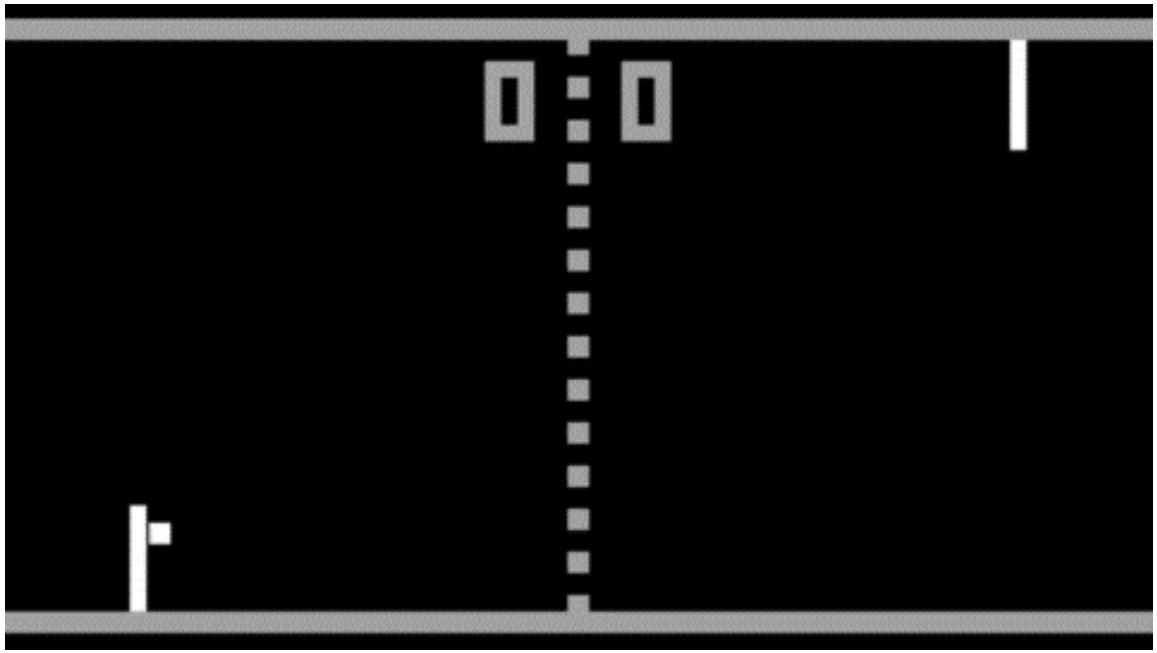


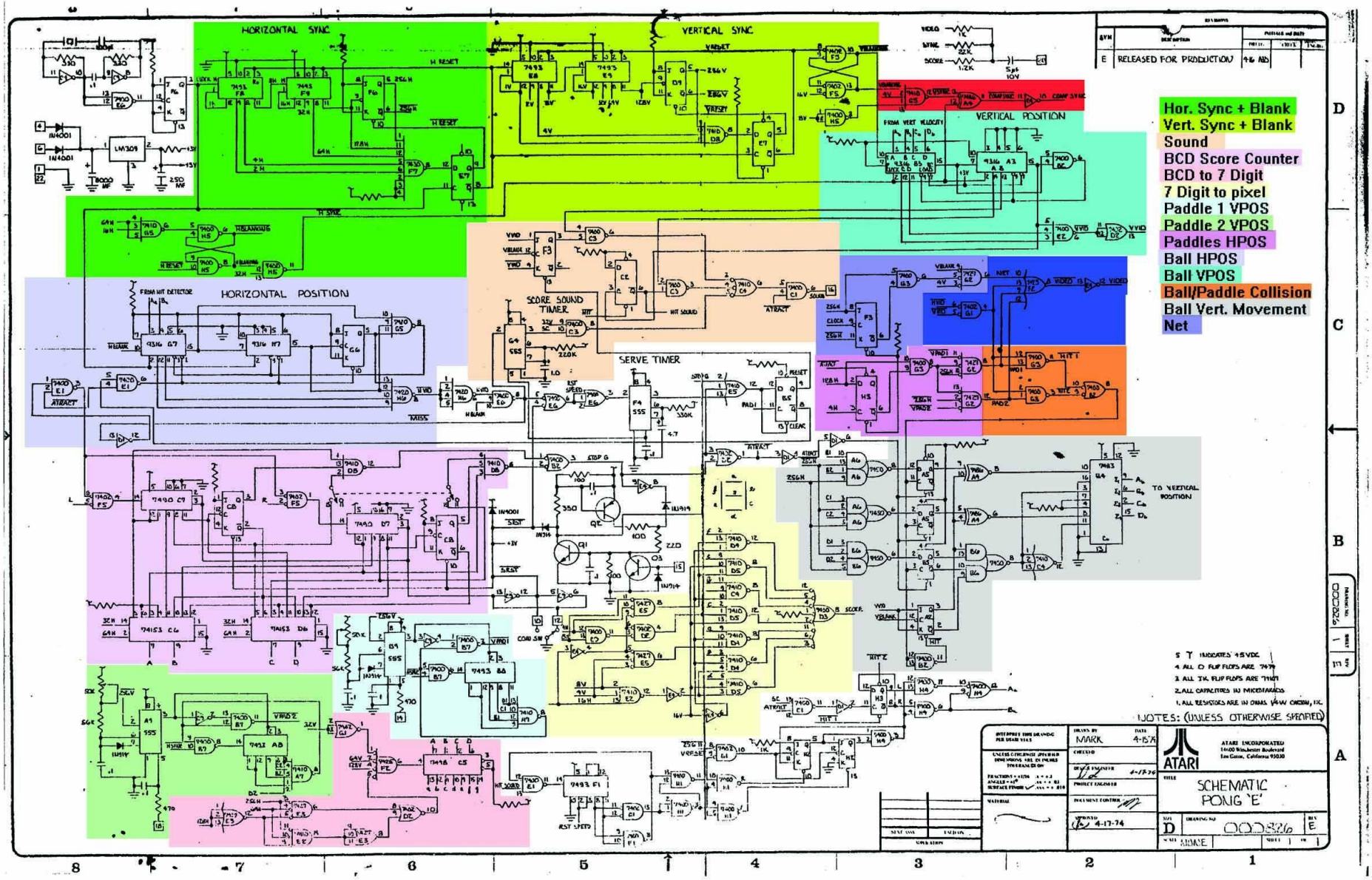

E. F. Moore. 1952. A simplified universal Turing machine. In Proceedings of the 1952 ACM national meeting (Toronto) (ACM '52), J. W. Forrester and R. W. Hamming (Eds.). ACM, New York, NY, USA, 50-54. DOI=http://dx.doi.org/10.1145/800259.808993

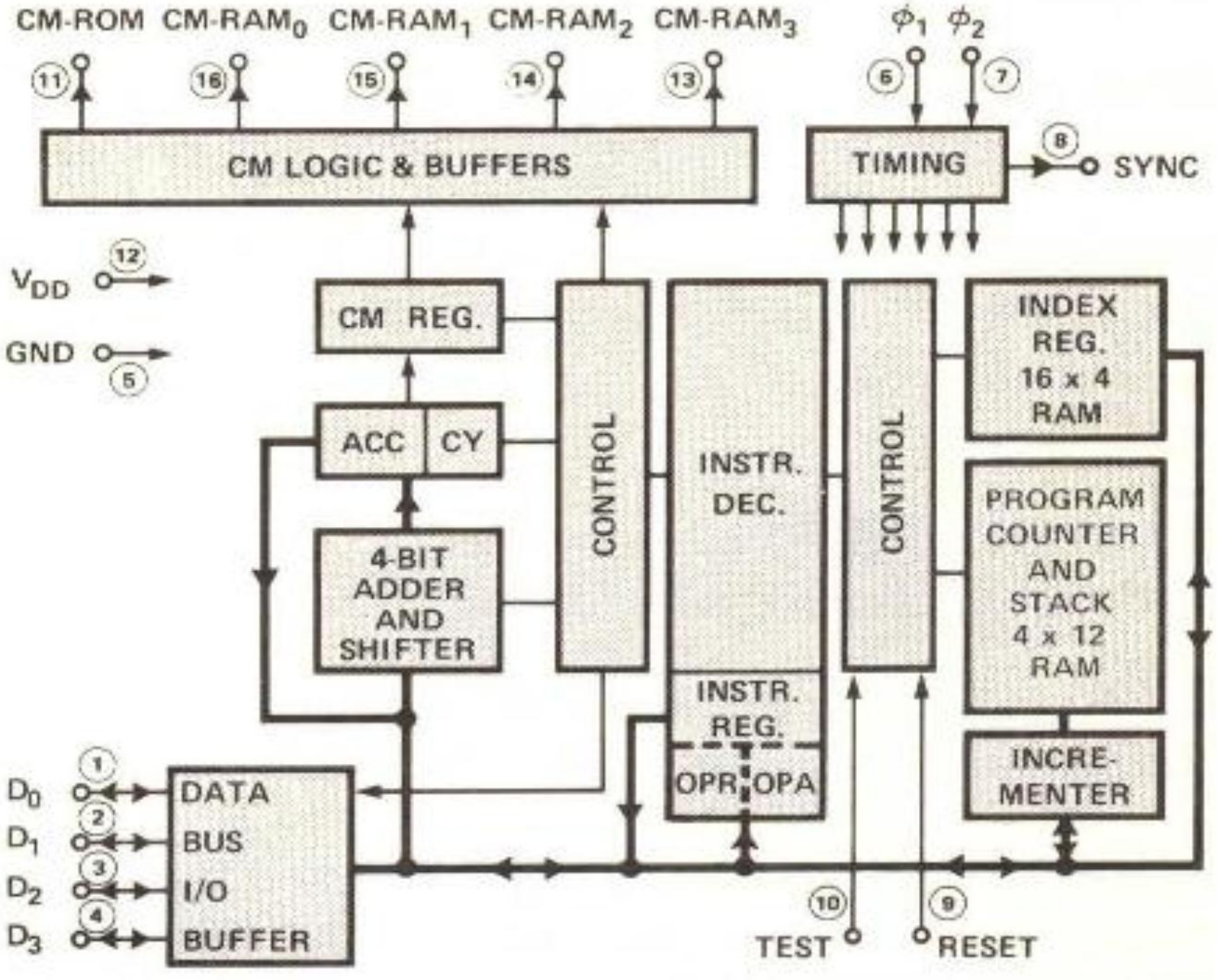
# Turing tax

- Alan Turing realised we could use digital technology to implement any computable function
- He then proposed the idea of a “universal” computing device – a *single* device which, with the right program, can implement any computable function *without further configuration*
- The “Turing Tax” is a term for the overhead (performance, cost, or energy) of universality in this sense
- That is, the performance difference between a special-purpose device and a general-purpose one
- **One of the fundamental questions of computer architecture is to how to reduce the Turing Tax**


PONG

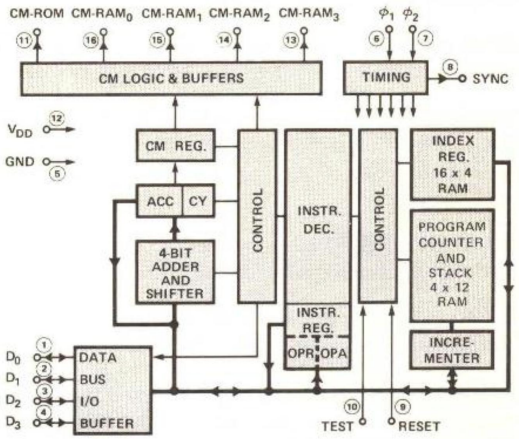


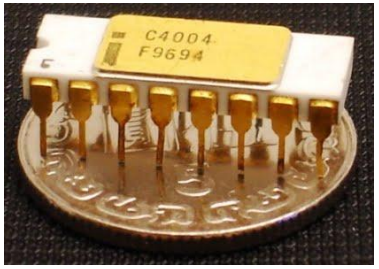

HORIZONTAL SYNC
VERTICAL SYNC
HORIZONTAL POSITION
VERTICAL POSITION
SCORE SOUND TIMER
SERVE TIMER
Hor. Sync + Blank
Vert. Sync + Blank
Sound
BCD Score Counter
BCD to 7 Digit
7 Digit to pixel
Paddle 1 VPOS
Paddle 2 VPOS
Paddles HPOS
Ball HPOS
Ball VPOS
Ball/Paddle Collision
Ball Vert. Movement
Net
TO VERTICAL POSITION
NOTES: (UNLESS OTHERWISE SPECIFIED)
E RELEASED FOR PRODUCTION
ATARI
SCHEMATIC
PONG 'E'
D
00D326
E



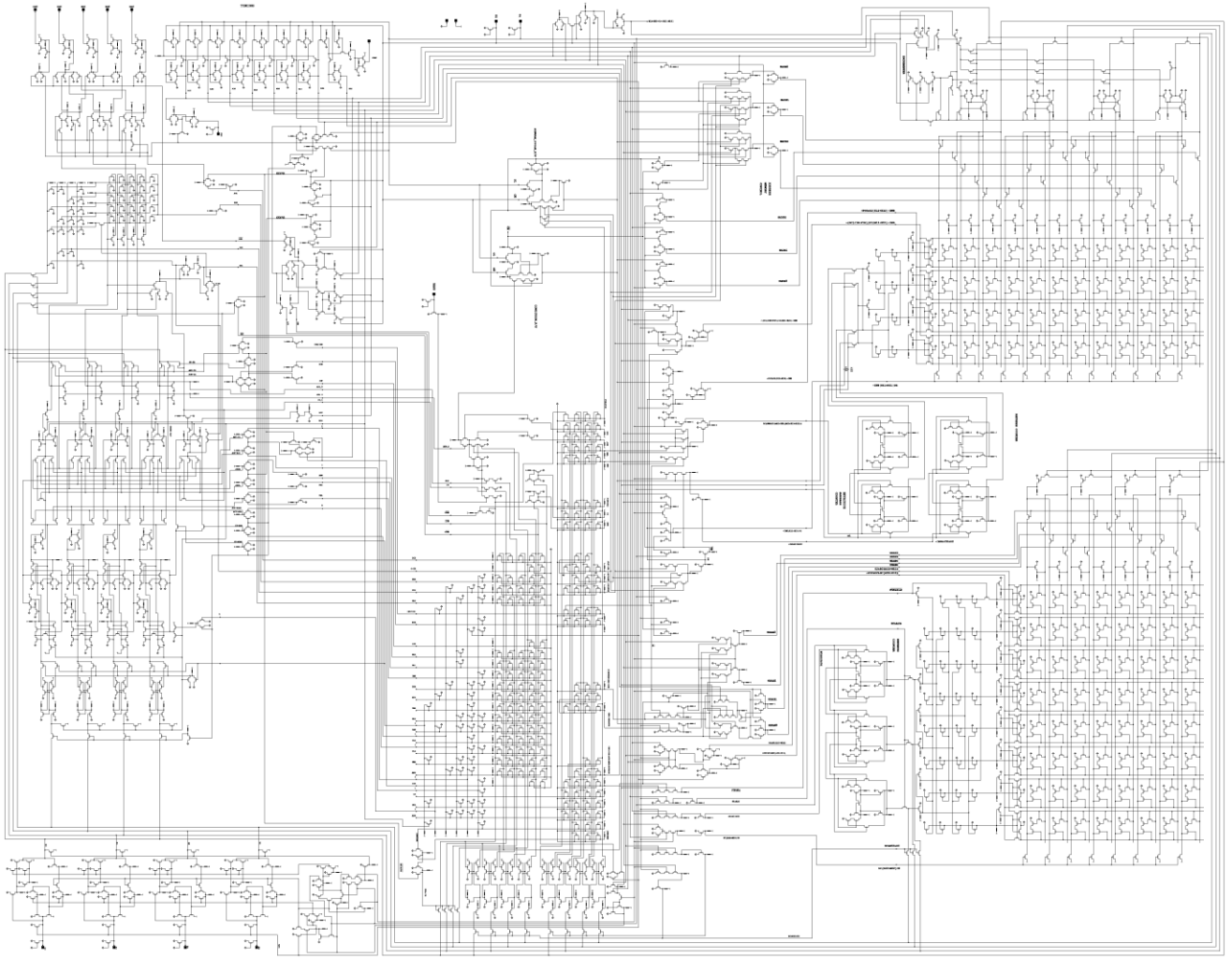

Block diagram for the first commercially-available microprocessor, Intel's 4004 (1971)

https://sites.google.com/site/intelcsclab/Home/intel-4004



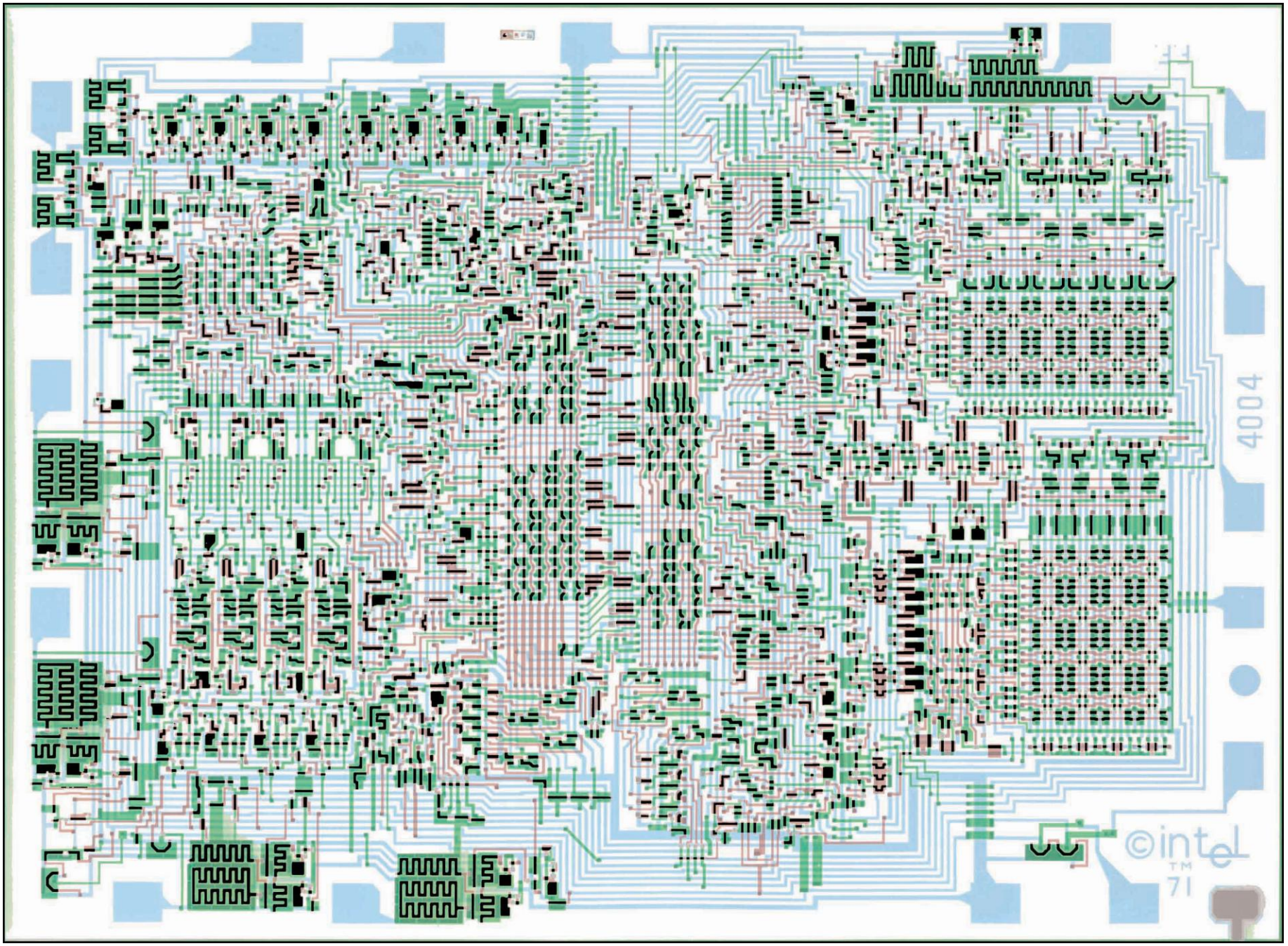




2300 transistors



Circuit diagram for the first commercially-available microprocessor, Intel's 4004 (1971)

https://www.4004.com/




Masks for Intel's 4004 microprocessor https://www.4004.com/

# Example: H.264 video encoder

| | Perf. (fps) | Area ($mm^2$) | Enrgy/frame (mJ) |
|---|---|---|---|
| Intel (720x480 SD) | 30 | 122 | 742 |
| Intel (1280x720 HD) | 11 | 122 | 2023 |
| ASIC | 30 | 8 | 4 |

- Intel's highly optimized, 2.8GHz Pentium 4 implementation of a 480p H.264 encoder versus a 720p HD ASIC.
- The second row presents Intel's SD data scaled to HD H.264.
- ASIC numbers have been scaled from 180nm to 90nm *(Hameed et al ISCA 2010)*

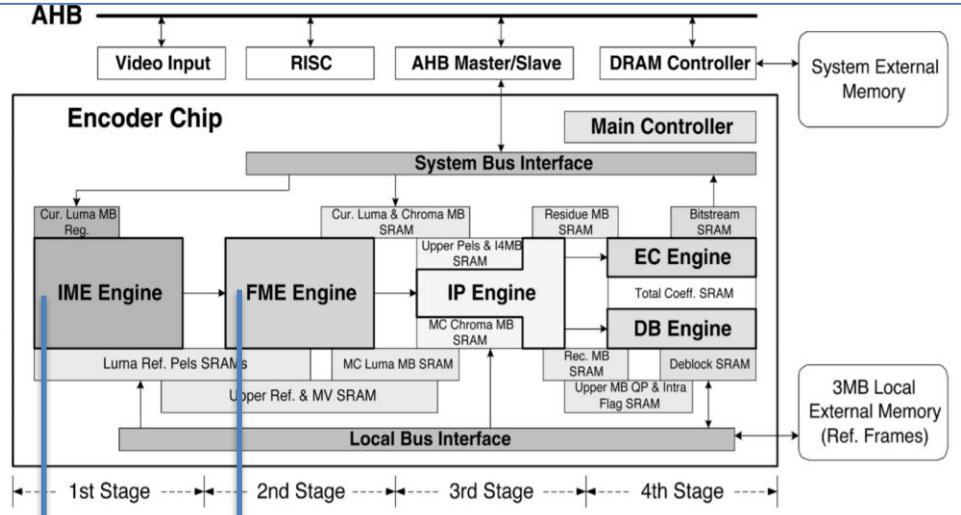



Fig. 2. Block diagram of the proposed H.264/AVC encoding system. Five major tasks, including IME, FME, IP, EC, and DB, are partitioned from the sequential encoding procedure and processed MB by MB in a pipelined structure.

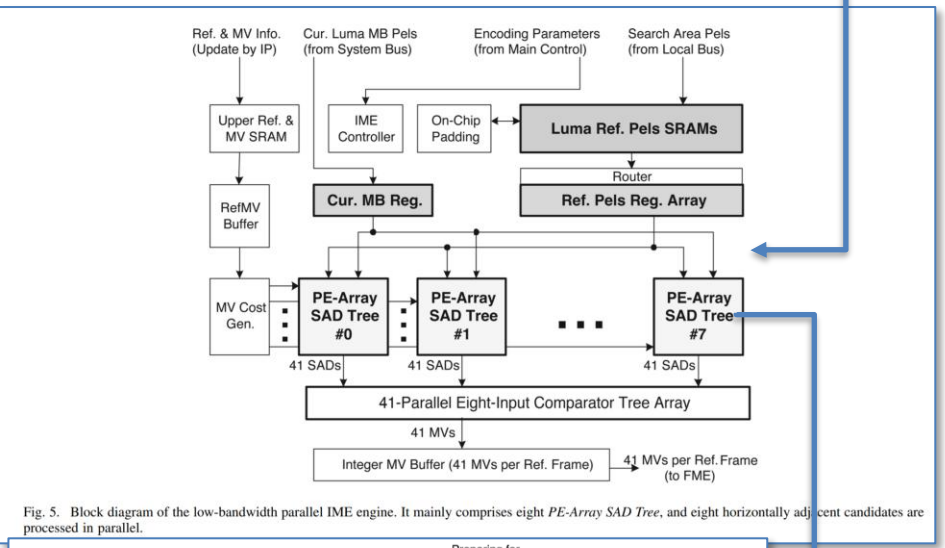



Fig. 5. Block diagram of the low-bandwidth parallel IME engine. It mainly comprises eight *PE-Array SAD Tree*, and eight horizontally adjacent candidates are processed in parallel.

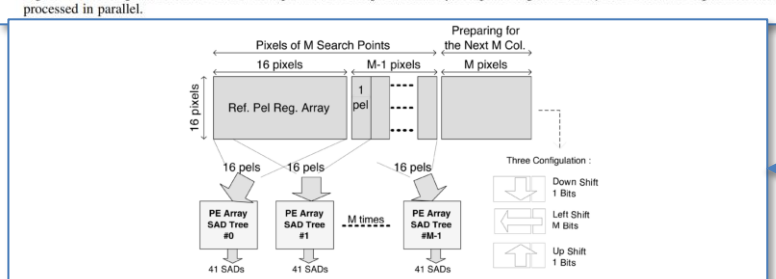



Fig. 6. M-parallel *PE-array SAD Tree* architecture. The inter-candidate DR can be achieved in both horizontal and vertical directions with *Ref. Pels Reg. Array*, and the on-chip SRAM bandwidth is reduced.

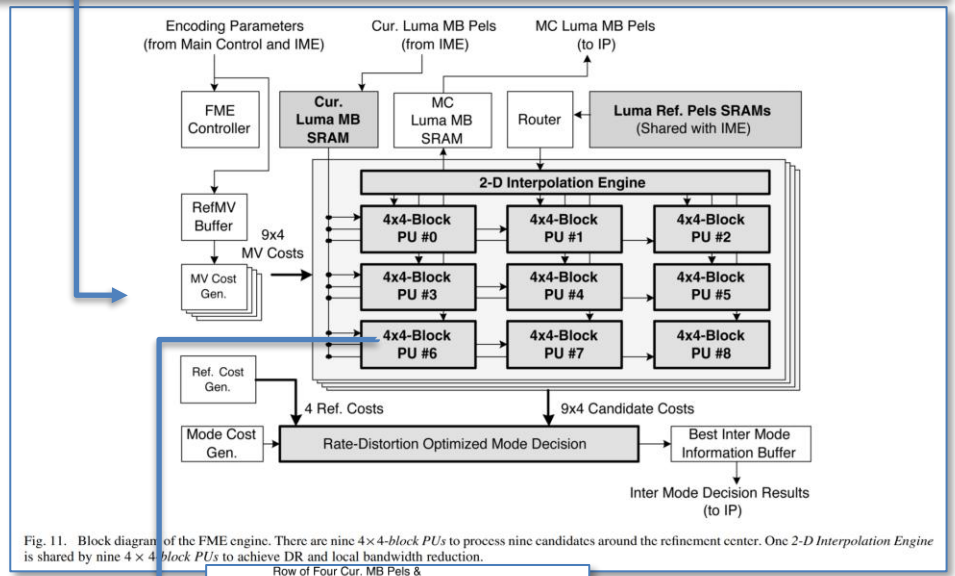



Fig. 11. Block diagram of the FME engine. There are nine 4×4-*block PUs* to process nine candidates around the refinement center. One *2-D Interpolation Engine* is shared by nine 4 × 4-*block PUs* to achieve DR and local bandwidth reduction.

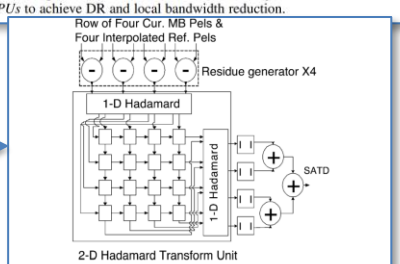



Fig. 12. Block diagram of 4 × 4-*block PU*. The *2-D Hadamard Transform Unit* is fully pipelined with *Residue Generators*.

*Tung-Chien Chen, Shao-Yi Chien, Yu-Wen Huang, Chen-Han Tsai, Ching-Yeh Chen, To-Wei Chen, and Liang-Gee Chen. 2006. Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder. IEEE Trans. Cir. and Sys. for Video Technol. 16, 6 (September 2006), 673–688. DOI:https://doi.org/10.1109/TCSVT.2006.873163*

- H.264 Is dominated by five stages
- Applied to a stream of macroblocks:

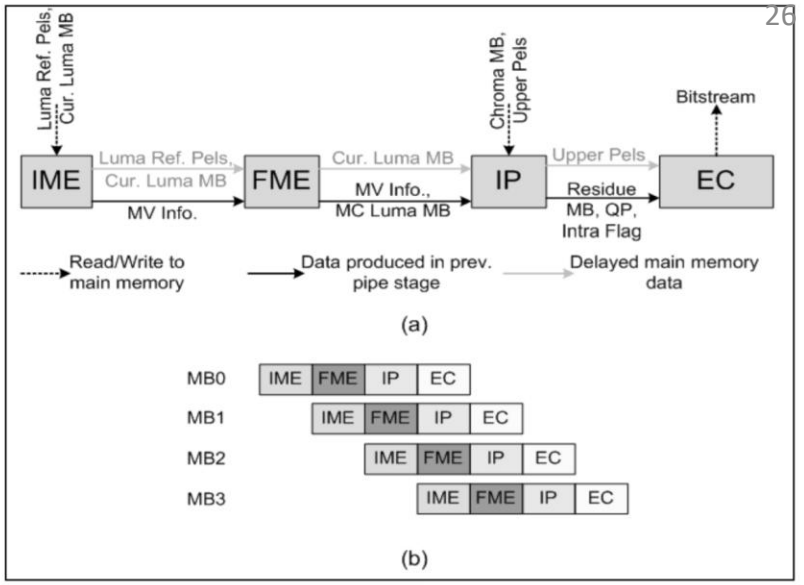
(i) IME: Integer Motion Estimation

(ii) FME: Fractional Motion Estimation

(iii) IP: Intra Prediction

(iv) DCT/Quant: Transform and Quantization and

(v) CABAC: Context Adaptive Binary Arithmetic Coding.



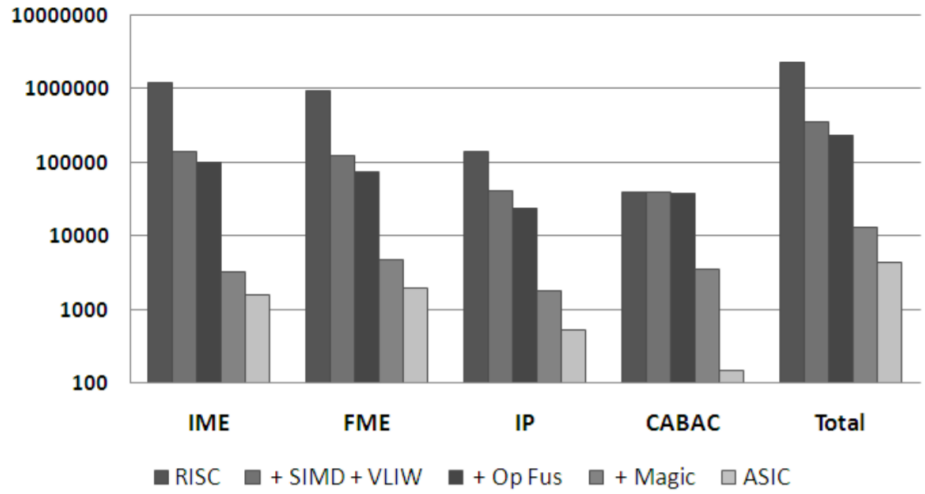




**Figure 2. Each set of bar graphs represents energy consumption (µJ) at each stage of optimization for IME, FME, IP and CABAC respectively. Each optimization builds on the ones in the previous stage with the first bar in each set representing RISC energy dissipation followed by generic optimizations such as SIMD and VLIW, operation fusion and ending with "Magic" instructions**

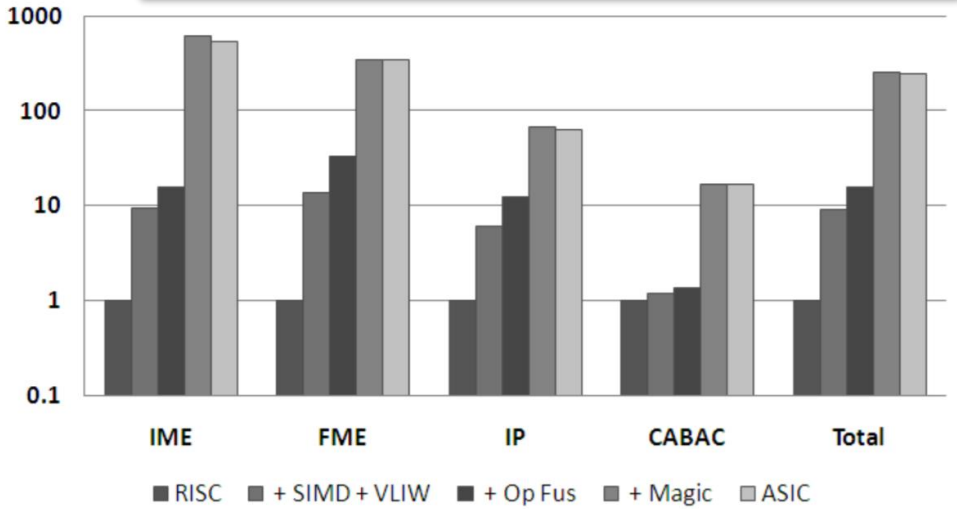
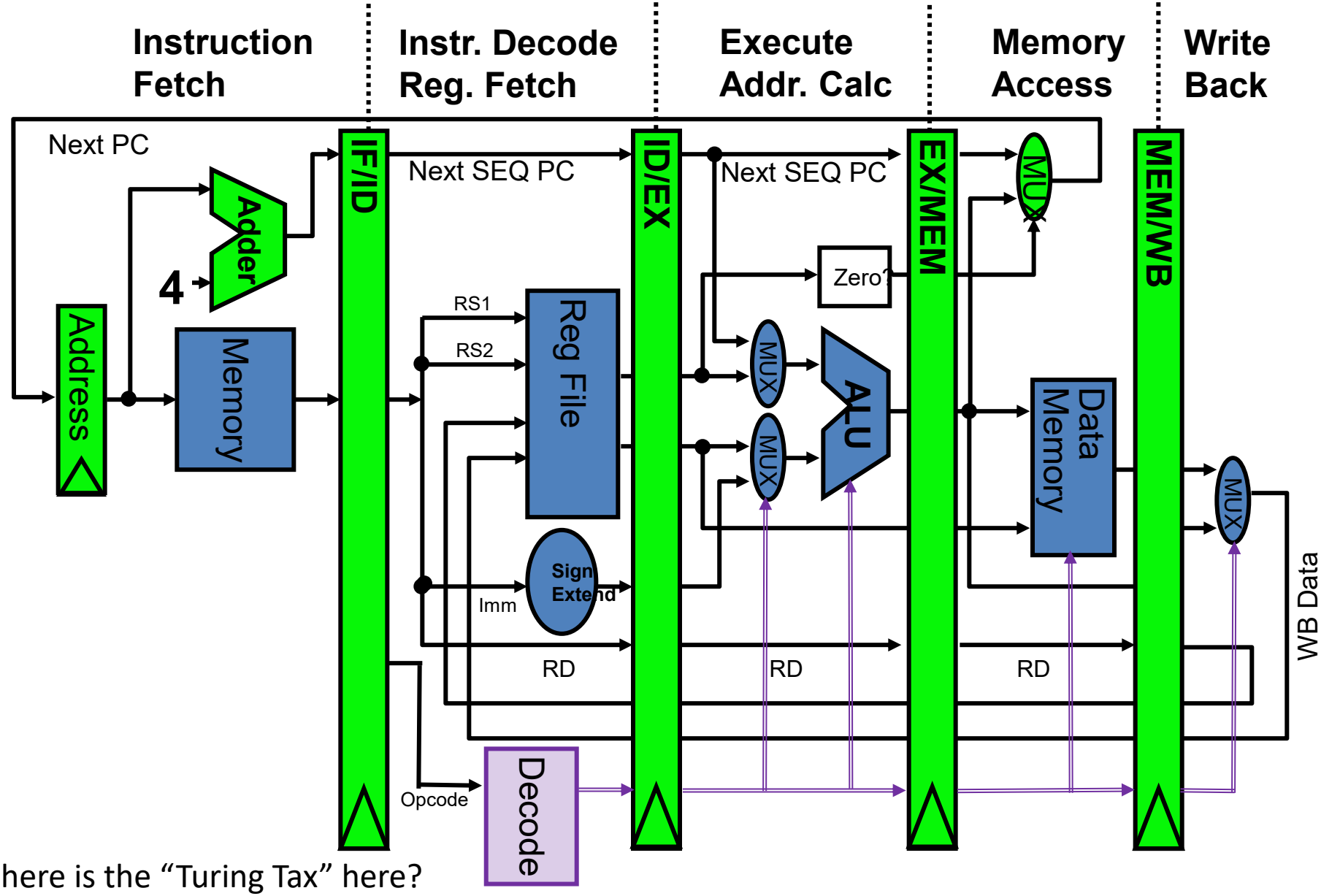



**Figure 3. Each set of bar graphs represents speedup at each stage of optimization. Each optimization builds on those of the previous stage with the first bar in each set representing RISC speedup, followed by generic optimizations such as SIMD and VLIW, then operation fusion and finally "Magic" instructions**

*Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. 2010. Understanding sources of inefficiency in general-purpose chips. In Proceedings of the 37th annual international symposium on Computer architecture (ISCA '10). Association for Computing Machinery, New York, NY, USA, 37–47. DOI:https://doi.org/10.1145/1815961.1815968*

# Pipelined MIPS Datapath with early branch determination




- Where is the "Turing Tax" here?
- That is – which bits are overhead due to the general-purpose nature of the processor, in contrast to a special-purpose digital design?

# Turing tax: instructions

- Instruction fetch
  - Store instructions
  - Fetch them
  - Decode them
  - Maintain PC
  - Handle branches

  - Predict branches
  - Handle branch mis-predictions

# Turing tax: data routing

- Forwarding is used to avoid stalls
- Forwarding is switched by multiplexors
- Which are determined by instruction decode

- We might not need all forwarding paths
- We might not need to switch them
- We might place the producer and consumer adjacently, so the wires can be shorter

# Turing tax: register access

- Instructions use registers to pass values from one operation to the next
- Each time a register is used, we have to look the value up in the register file

- In a special-purpose machine, we'd use a piece of wire!

# Turing tax: configurable ALU

- In our MIPS pipeline, the ALU function is controlled by a signal derived from decoding the instruction
- The ALU is a multipurpose unit – that can add, subtract, multiply etc

- In a special-purpose design we would only have the units we need
- and we'd have just the right number of each kind

# Turing tax: avoidance?

## What can we do to avoid the Turing Tax?

# Caches are “Turing Tax”

# Discuss!

# The Turing Tax is irrelevant for most applications

# Discuss!