in *Berichte der Gesellschaft für Mathematik und Datenverarbeitung* **63** (1972), Part 1, 32 pages. English translation, **106** (1976), 7–20.

[ZU 45] K. Zuse, *Der Plankalkül*, manuscript prepared in 1945. Published in *Berichte der Gesellschaft für Mathematik und Datenverarbeitung* **63** (1972), Part 3, 285 pages. English translation of all but pages 176–196, **106** (1976), 42–244.

[ZU 48] K. Zuse, "Über den Allgemeinen [sic] Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben," *Archiv der Mathematik* **1** (1949), 441–449.

[ZU 59] K. Zuse, "Über den Plankalkül," *Elektronische Rechenanlagen* **1** (1959), 68–71.

[ZU 72] Konrad Zuse, "Kommentar zum Plankalkül," *Berichte der Gesellschaft für Mathematik und Datenverarbeitung* **63** (1972), Part 2, 36 pages. English translation, **106** (1976), 21–41.

Addendum

Two other languages, Transcode and PACT I, deserve to be part of the story as well, so they appear in Table 1 above although they were unfortunately missed by the authors when we first compiled this history.

Transcode was developed for the FERUT computer, a clone of the Ferranti Mark I that was installed at the University of Toronto in 1952. The authors of this language, J. N. P. Hume and B. H. Worsley, devised a way to cope with FERUT's awkward two-level storage that was more efficient than the AUTOCODE approach being taken independently by Brooker in Manchester, because they expected Transcode programmers to be aware of FERUT's overall characteristics. To perform the TPK algorithm they might have punched the following codes onto a paper tape:

000	`INST 019`	Inputs the following 19 Instructions.
001	`READ 001.0 000.0 X00.0`	Copies `DRUM` location 1 into X-page.
002	`ADDN C01.0 C01.0 Z01.0`	Sets initial value of $i = 10 - j$ to $5 + 5$.
003	`BSET 000.5 000.0 000.0`	Sets initial value of B_5 to 0.
004	`LOOP 011.0 000.6 000.0`	Prepares to loop 11 times, using B_6.
005	`KOMP X11.5 C02.0 Z02.0`	Places $\lvert a_i\rvert - 0$ in Z02.
006	$\frac{1}{2}$`QRT Z02.0 000.0 Z02.0`	Forms $\lvert a_i\rvert^{1/2}$.
007	`MULT X11.5 X11.5 Z03.0`	Places $a_i \cdot a_i$ in Z03.
008	`MULT X11.5 Z03.0 Z03.0`	Forms $a_i \cdot a_i \cdot a_i$.
009	`MULT C01.0 Z03.0 Z03.0`	Forms $5a_i \cdot a_i \cdot a_i$.
010	`ADDN Z02.0 Z03.0 Z02.0`	Forms $f(a_i)$.

011	`SUBT C03.0 Z02.0 Z03.0`	Forms $400 - f(a_i)$.
012	`TRNS 014.0 000.0 Z03.0`	Transfers control if $400 - f(a_i) \geq 0$.
013	`OVER C04.0 000.0 Z02.0`	Enters 999 in place of $f(a_i)$.
014	`PRNT 002.2 010.0 Z01.0`	Prints answers.
015	`SUBT Z01.0 C05.0 Z01.0`	Adjusts i to its next value.
016	`INCB 000.5 003.0 000.0`	Adjusts B_5 to its next value.
017	`TRNS 005.0 000.6 000.0`	Ends loop, adjusting B_6.
018	`HALT 000.0 000.0 000.0`	Programs a stop.
019	`QUIT 000.0 000.0 000.0`	Terminates the Instructions.
020	`CNST 5++ 1+100- 4+2+ 999+2+ 1++ "`	Specifies the five constants.
021	`NUMB` $a_0\ a_1 \ldots\ a_{10}$ `"`	Specifies the input data.
022	`DRUM 001`	Stores block 1 of numerical data.
023	`ENTR`	Begins compilation and execution.

The variables in Transcode, other than those in B registers, were floating-point numbers `X01`, `X02`, ..., `Y01`, `Y02`, ..., `Z01`, `Z02`, ..., consisting of three 20-bit words each, namely a 40-bit signed fraction and a signed 20-bit exponent. But Transcode programmers didn't have to deal with binary notation; for example, the five constants `C01` $= 5$, `C02` $= 0$, `C03` $= 400$, `C04` $= 999$, `C05` $= 1$ in the program above are specified on line *020* in decimal form as ⟨mantissa⟩ ⟨sign⟩ ⟨exponent⟩ ⟨exponent sign⟩. A constant like $-.0073$ would be '`73-3-`'; and the same conventions applied to input data in a `NUMB` specification such as line *021*. Notice that they used 1^{-100} for the constant zero(!). The `PRNT` command on line *014* would output `Z01` and `Z02` to a line on FERUT's typewriter, showing both numbers in floating-point notation with ten significant digits.

Variables were stored backwards in memory, so that the address of the first word of `X02` was 3 locations *less* than the address of `X01`. The main loop of this program, governed by instructions *003*, *004*, *016*, and *017*, is performed for eleven index register settings $(B_5, B_6) = (0, 30)$, $(3, 27)$, ..., $(30, 0)$; thus `X11.5` is `X11` the first time, then `X10`, ..., then `X01`. Meanwhile variable $i =$ `Z01` steps through the values 10, 9, ..., 0, because of instructions *002* and *015*.

Programmers could say `KOPY` at the end, following `ENTR`; then the compiled instructions would be punched on paper tape, for subsequent use as a subroutine in other programs. Since paper tape code was equivalent to teletype code, programs could also be transmitted "online" to Toronto from remote sites in Canada. [For further details, see J. N. P. Hume and Beatrice H. Worsley, "Transcode: A system of automatic coding for FERUT," *Journal of the Association for Computing Machinery* **2** (1955), 243–252; J. N. Patterson Hume, "Development of systems software for the Ferut computer at the University of Toronto, 1952 to 1955," *IEEE Annals of the History of Computing* **16**, 2 (Summer 1994), 13–19.]