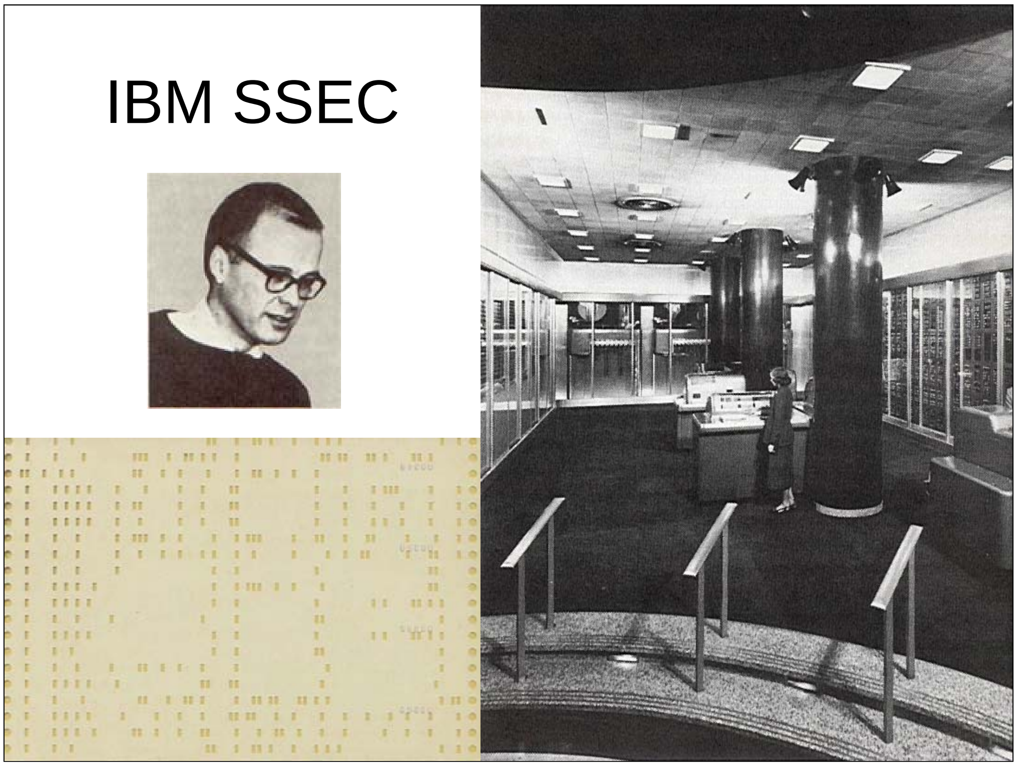
A Tour of a Software Collection: IBM 704 FORTRAN

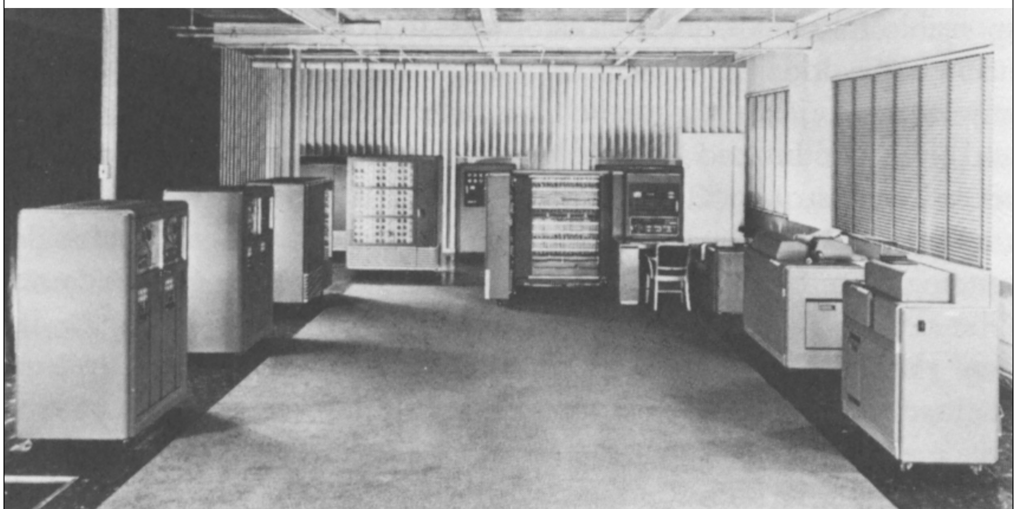
Paul McJones
23 June 2005
Board Retreat

The next few slides are intended to give you an impression of what you can learn studying the materials on the SCC's FORTRAN web site.

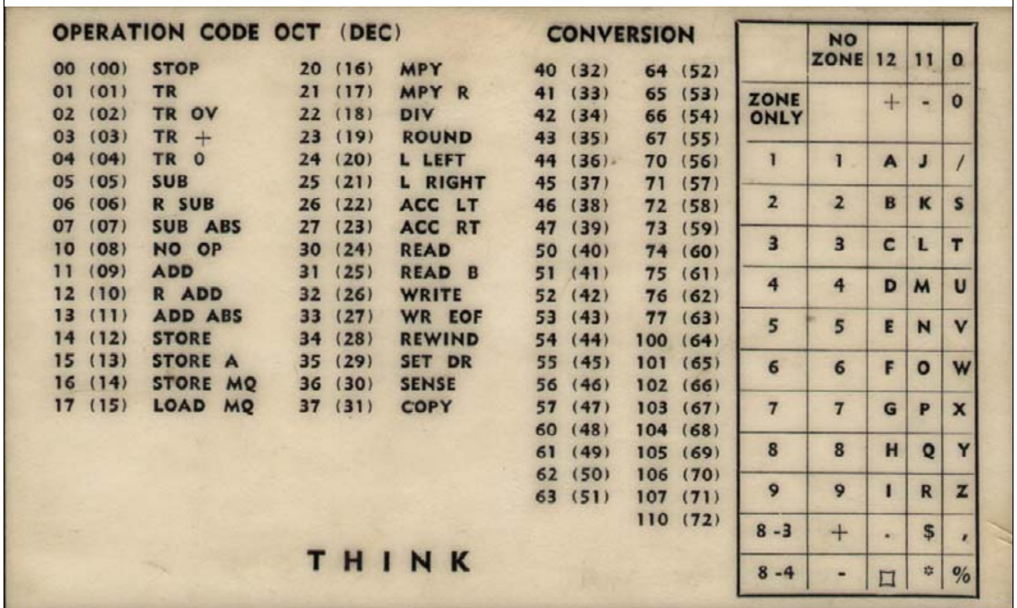


A few months before I was born, John Backus walked into IBM off the street, and was offered a job programming the SSEC. He accepted the job, and was soon immersed in the complexities of this electromechanical monster. One pitfall he told me about years later: accidentally gluing the long paper tape into a Moebius strip.

IBM 701



A few years later, IBM completed the 701, a true stored-program computer, and Backus, who considered himself lazy, considered how to ease the programming task.



OPERATION CODE OCT (DEC)					
00	(00)	STOP	20	(16)	MPY
01	(01)	TR	21	(17)	MPY R
02	(02)	TR OV	22	(18)	DIV
03	(03)	TR +	23	(19)	ROUND
04	(04)	TR 0	24	(20)	L LEFT
05	(05)	SUB	25	(21)	L RIGHT
06	(06)	R SUB	26	(22)	ACC LT
07	(07)	SUB ABS	27	(23)	ACC RT
10	(08)	NO OP	30	(24)	READ
11	(09)	ADD	31	(25)	READ B
12	(10)	R ADD	32	(26)	WRITE
13	(11)	ADD ABS	33	(27)	WR EOF
14	(12)	STORE	34	(28)	REWIND
15	(13)	STORE A	35	(29)	SET DR
16	(14)	STORE MQ	36	(30)	SENSE
17	(15)	LOAD MQ	37	(31)	COPY

CONVERSION			
40	(32)	64	(52)
41	(33)	65	(53)
42	(34)	66	(54)
43	(35)	67	(55)
44	(36)	70	(56)
45	(37)	71	(57)
46	(38)	72	(58)
47	(39)	73	(59)
50	(40)	74	(60)
51	(41)	75	(61)
52	(42)	76	(62)
53	(43)	77	(63)
54	(44)	100	(64)
55	(45)	101	(65)
56	(46)	102	(66)
57	(47)	103	(67)
60	(48)	104	(68)
61	(49)	105	(69)
62	(50)	106	(70)
63	(51)	107	(71)
		110	(72)

	NO ZONE	12	11	0
ZONE ONLY		+	-	0
1	1	A	J	/
2	2	B	K	S
3	3	C	L	T
4	4	D	M	U
5	5	E	N	V
6	6	F	O	W
7	7	G	P	X
8	8	H	Q	Y
9	9	I	R	Z
8 -3	+	.	$	,
8 -4	-	□	*	%

THINK

The machine had no index register or floating-point arithmetic, as indicated by this quick reference card that once belonged to Roy Nutt, who worked for United Aircraft at the time.

for
Jan Lee

IBM 701 SPEEDCODING AND OTHER AUTOMATIC-
PROGRAMMING SYSTEMS

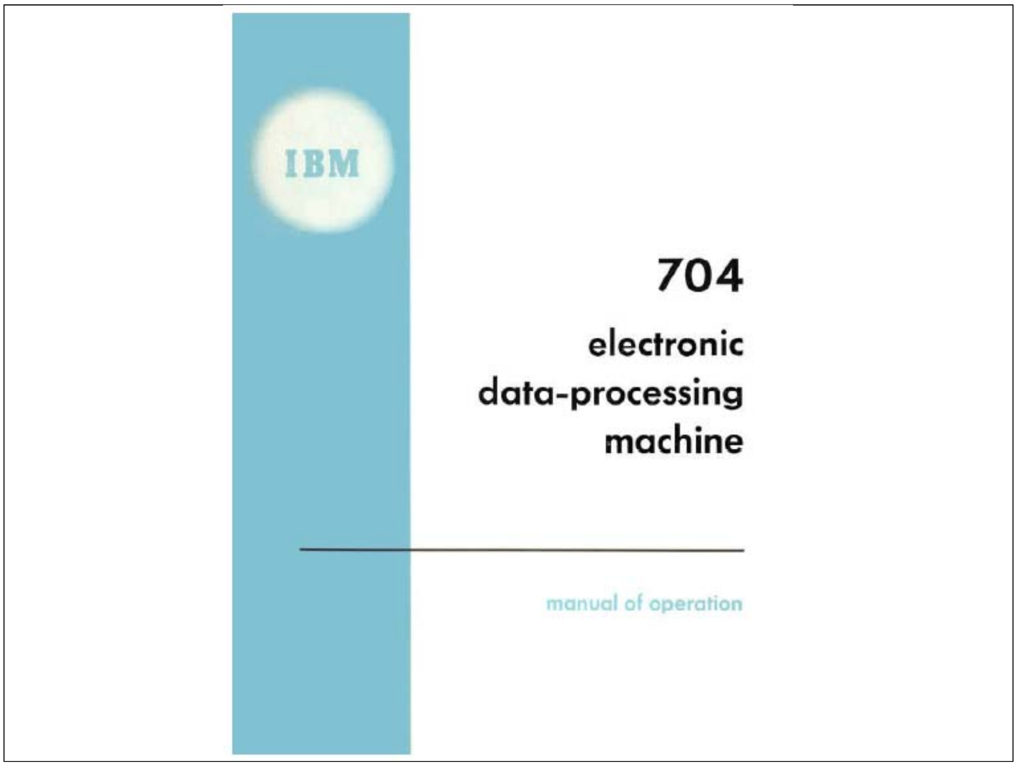
John W. Backus and Harlan Herrick
International Business Machines Corporation, New York, New York

Before the discussion of IBM's 701 Speedcoding System, it is interesting to consider the entire area of automatic programming for large-scale calculators to obtain a perspective as to the position of such a system in this field. The basic purposes of automatic-programming systems are primarily the following:

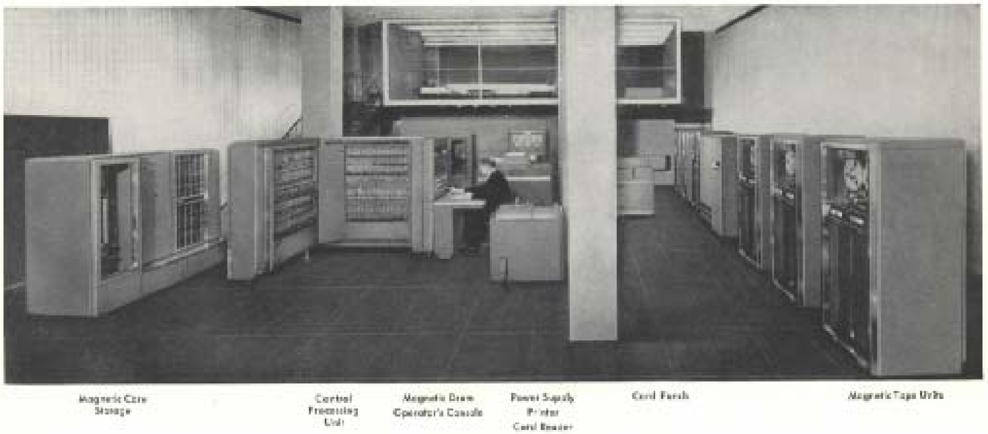
1. to provide operations which are not part of a machine's repertoire as operations of the system,
2. to provide operations which are part of a machine's repertoire in a more convenient form.

In other words, it is desired to supply the programmer with a more convenient language in which to designate a problem than that now used by the machine.

So Backus and his colleague Harlan Herrick came up with an interpretive system called Speedcoding that essentially provided a virtual machine that was more capable but slower than the real 701.



In the mean time, IBM was learning from the 701 and building a much more reliable and high performance scientific computer, the 704.



IBM 704 ELECTRONIC DATA-PROCESSING MACHINES

It had core memory to the 701's Williams tubes, and …

All of the real work is done in the central processing unit; that is, all additions, subtractions, multiplications, etc. are done in the special registers of the central processing unit. In addition to standard arithmetic, the 704 has instructions which will perform logical arithmetic for increased flexibility in doing complex problems. Also in the central processing unit are three index registers for automatic counting and effective address modification.

An important feature on the 704 is a complete set of instructions which will perform floating-point arithmetic. This manual includes a complete description of floating-point numbers and the special floating-point instructions (such as floating add, subtract, multiply, divide or halt, and divide or proceed) needed to manipulate data in this form.

… both index registers and floating-point. Backus realized that he needed to move beyond Speedcoding to make an impact on programmer productivity.

There was lots of pent-up demand for the 704, as evidence by this photograph of customers learning about the machines they would soon be taking delivery of.

IBM DEPARTMENT OF EDUCATION
Customer Administrative School - Class No. 941
May 3 to 6, 1954

PICTURE IDENTIFICATION

ROW I

1. L. S. Crandall
2. S. Crossman
3. H. R. J. Grosch
4. D. Blum
5. E. G. Kogbetliantz
6. L. H. Amaya
7. R. E. Porter
8. F. C Malone
9. G. T. Hunter
10. L. Walters

ROW II

1. W. C. Davison
2. W. F. McClelland
3. P. Fagg
4. W. P. Simonet
5. H. S. Wolanski
6. G. M. Amdahl
7. H. Tillitt
8. D. W. Pendery
9. S. Fernbach

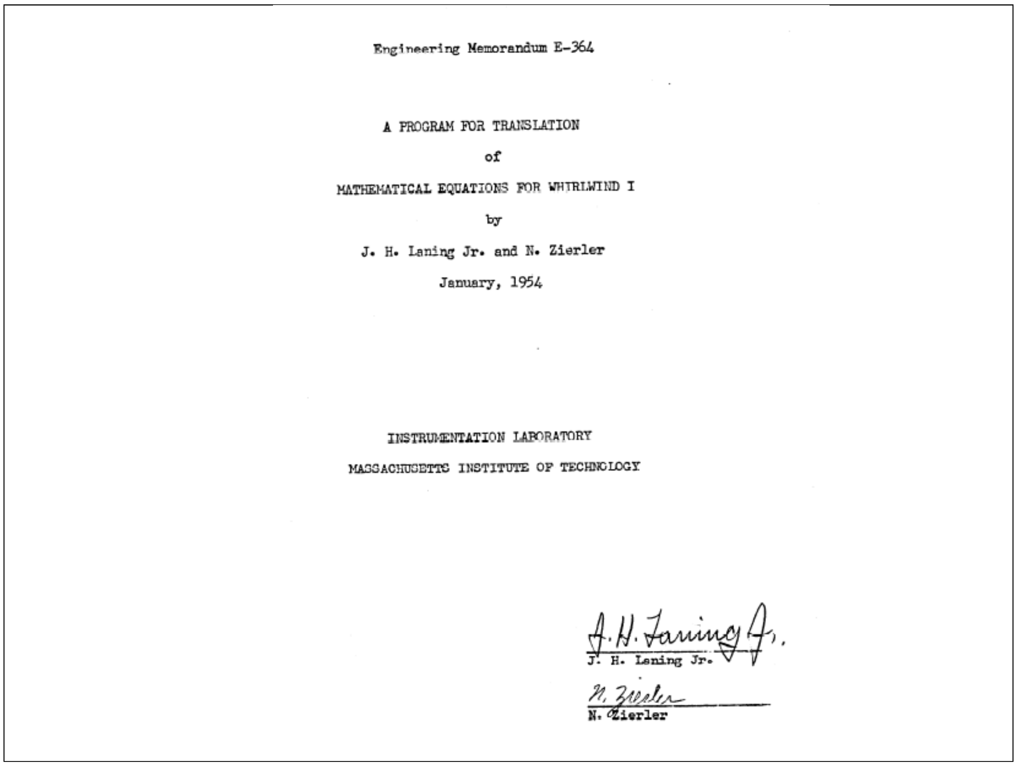
ROW III

1. J. Backus
2. D. H. Furth
3. W. B. Dobrusky
4. J. T. Horner
5. D. E. Hart
6. H. G. Kolsky
7. F. G. Shuman
8. J. F. Stockman
9. B. Carlson
10. J. D. Madden
11. I. C. Liggett
12. J. A. Strong

ROW IV

1. J. C. McPherson
2. W. L. Williams
3. B. Oldfield
4. J. Griffith
5. G. W. Petrie
6. W. A. Ramshaw
7. J. B. Hughes
8. J. Chancellor
9. R. L. Johnson
10. D. W. Sweeney
11. P. Armer

Note John Backus, Gene Amdahl, Herb Grosch, and Harwood Kolsky, among others.


Engineering Memorandum E-364

A PROGRAM FOR TRANSLATION

of

MATHEMATICAL EQUATIONS FOR WHIRLWIND I

by

J. H. Laning Jr. and N. Zierler

January, 1954

INSTRUMENTATION LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

J. H. Laning Jr.
J. H. Laning Jr.

N. Zierler
N. Zierler

Meanwhile, many people were thinking about how to ease the task of programming, including Laning and Zierler at MIT's Whirlwind project. They wrote an interpreter ...

10. Examples

Suppose it is desired to print a table of values of cosine x for x = 0,.1,.2,...,1 radian and that we decide to use up to the x^{10} term in the power series for cosine x. This could be accomplished by the following program:

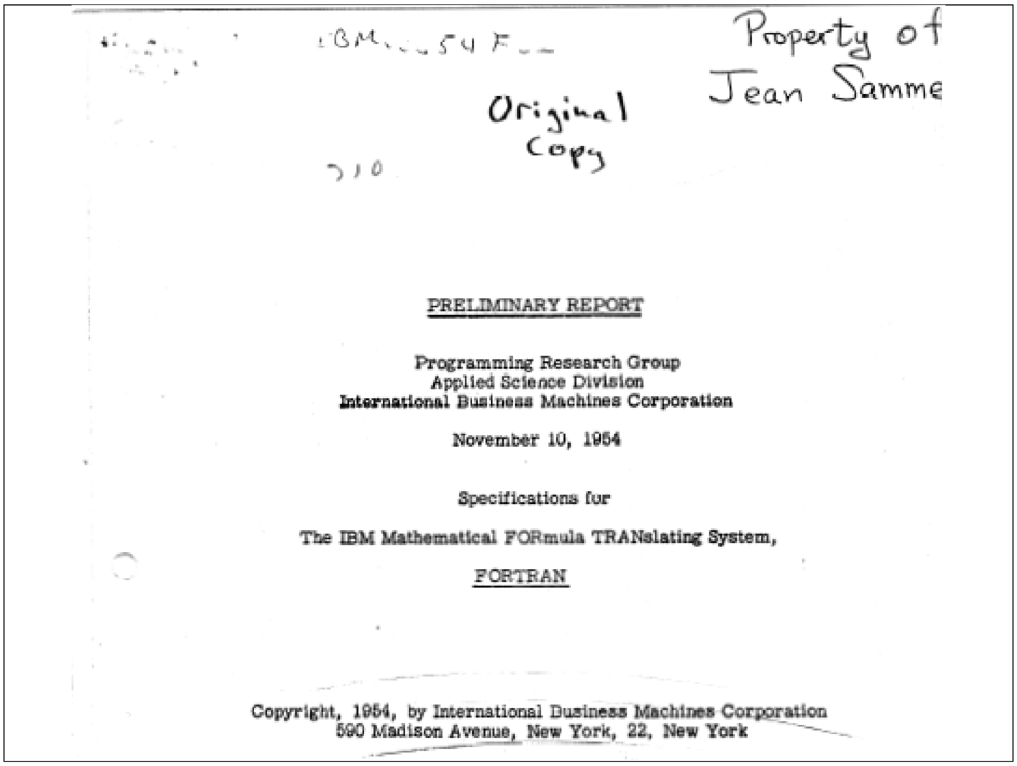
```
        x = 0,
1       z = 1 - x^2/2 + x^4/2·3·4 - x^6 2·3·4·5·6
            + x^8/2·3·4·5·6·7·8 - x^10/2·3·4·5·6·7·8·9·10,
        PRINT x, z.
        x = x + .1,
        e = x - 1.05,
        CP 1,
        STOP
```

We might prefer an iterative scheme to produce the same result. The following scheme has the advantage that the number of terms in the series may be adjusted by changing a single number:

```
        x = 0,
1       y = 10,
        z = 1,
2       z = 1 - z x^2/y (y - 1),
        y = y - 2,
        e = 1 - y,
        CP 2,
        PRINT x, z.
        x = x + 0.1,
        a = x - 1.05,
        CP 1,
        STOP
```

13

… that Backus (who with Herrick independently had a similar idea) believes “was the world's first operating algebraic compiler, a rather elegant but simple one.”



Property of
Jean Samme

Original
Copy

PRELIMINARY REPORT

Programming Research Group
Applied Science Division
International Business Machines Corporation

November 10, 1954

Specifications for

The IBM Mathematical FORmula TRANslating System,

FORTRAN

Backus went to his boss Cuthbert Hurd, and proposed to build a compiler, not an interpreter, for an algebraic language to be called FORTRAN.

15. DESIRABLE TECHNIQUES TO USE IN PROGRAMMING A PROBLEM TO BE CODED BY FORTRAN

Although the FORTRAN system is being designed to produce a correct program from a correct meaningful set of FORTRAN formulas and although the programmer will invariably discover many possible formulations of the same problem, the use of certain techniques will, of course, result in more efficient 704 programs.

A. REPRESENTATION OF COMPLICATED EXPRESSIONS

In translating a single arithmetic formula, the FORTRAN system will permute the operations indicated in the expression on the right wherever this is permissible in order to minimize the number of STORE instructions which will be required in the resulting 704 program. Thus a × b × c /d/e would be permuted to a/d×b/e×c. However, any order of computation which is specified by use of parenthesis will be followed. Furthermore, if certain portions of an expression are identical to certain other portions of the same expression (all in the same formula), the system will recognize this and avoid duplicate calculations. To enable the FORTRAN system to recognize duplications of various subexpressions in an expression on the right side of an arithmetic formula, it will only be necessary to enclose duplicated subexpressions where they appear as part of a term in the expression. Where duplicated subexpressions occur as complete terms, it will not be necessary to enclose the term in parentheses. Furthermore, if the duplicated subexpression is a function which appears in several places with the same argument, it will not be necessary to enclose the function in parentheses even though it may be a portion of a term. Thus the following expression:

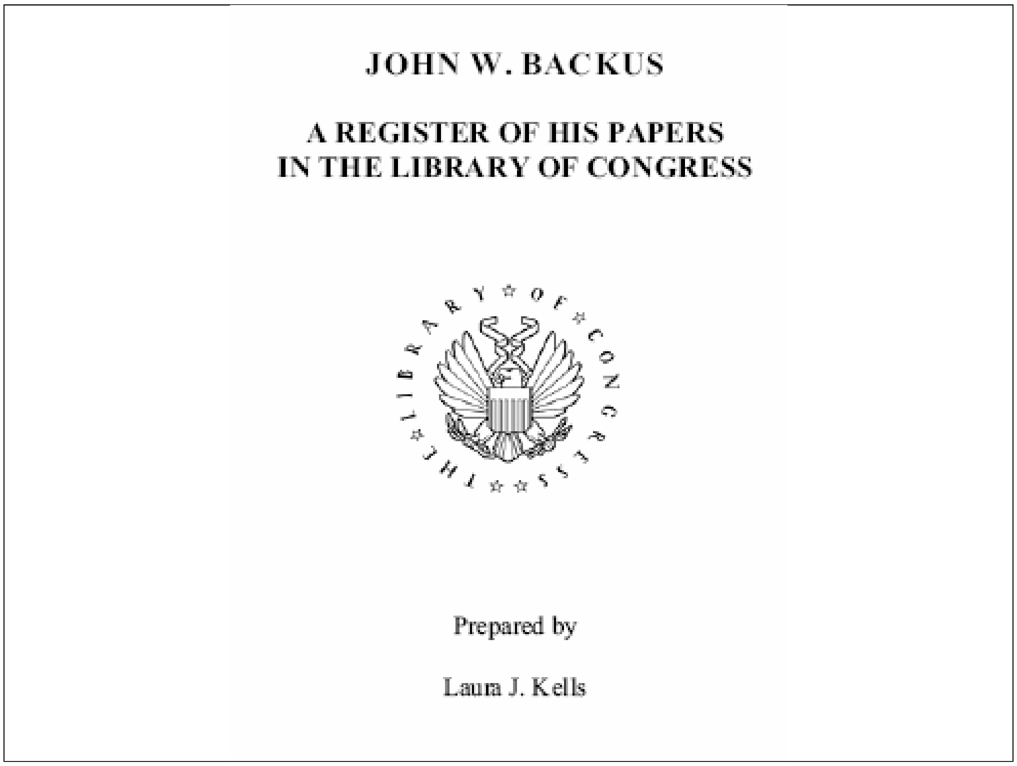
a × b × c ×(a ×b×c+e×cos(a))/(a×b×e+f×cos(a)) +sqrt (a×b×e+f×cos(a))

may be written in the following form to avoid duplicate calculations:

((a×b)×c)×((a×b)×c+e×cos(a))/((a×b)×e+f×cos(a)) + sqrt((a×b)×e+f×cos(a))

In general then, if a complicated expression is involved in a problem, it is best not to introduce new dependent variables to represent portions of the complicated expression and then to represent the complicated expression as an expression involving the new dependent variables. Adherence to this principle allows the FORTRAN system to carry out the maximum amount of optimization.

From the beginning, he realized that the compiler would need to generate very efficient machine code if he was to convince the aerospace companies and radiation laboratories to use it.


JOHN W. BACKUS

A REGISTER OF HIS PAPERS
IN THE LIBRARY OF CONGRESS

THE LIBRARY OF CONGRESS

Prepared by

Laura J. Kells

Unfortunately we don't have any email archives from the mid 1950s, but Backus donated his papers to the Library of Congress …

Items for the Library of Congress

Item No.	Name of item	Box #/	Description/Comments
			many of the objectives for Fortran (as discussed in item # 18).
105	*Automatic coding for digital computers* by Grace Murray Hopper, no date	2	
106	Roy Nutt's Correspondence (in and out) 5/58 to 4/55	2	Lots of stuff about debugging Fortran, back and forth between Roy and Fortran group
107	Roy Nutt's trips to NY to work with Fortran group + more correspondence with us	2	
108	Notes & Manual for Speedcoding 1 & 2	2	
109	*Description of source language additions to the Fortran II system* this describes what became known as Fortran III. It is the work of Irv Ziller.	2	Fortran III allowed users to insert symbolic 704 instructions into a Fortran source program
110	Letter from me to Franz Ross, Applied Science Publications Group announcing a version of Fortran I for machines with a larger core storage unit (8192 words or more). The 1st version would run with 4096 words	2	May 7, 1957. This letter also announces a Fortran editing program, FNEDT1
111	*Preliminary report: Proposed specifications for Fortran II for the 704* 6/28/57 see item # 76	2	
112	Note from David Sayre re the origin of the "function statement" in Fortran I	2	
113	Letter from me to John Greenstadt (responsible for distributing programs to SHARE) announcing the availability of FN1 (1st version of Fortran) on binary cards + an addenda to the manual + preliminary operator's manual. February 8, 1957	2	It soon turned out to be impossible to punch the binary cards, so this distribution did not take place. See item # 114
114	Letter from me to John Greenstadt announcing the availability of Fortran 4-1-4-1 (requiring at least 4096 words of core storage) to be distributed on tapes.	2	
115	Programming Research Dept memos: (a) Tel extensions 3/19/57, (b) Job classifications 3/20/57, (c) ditto 8/15/58	2	
116	"Pro Res" 1957 – Dept newsletter produced by secretary Rosemarie Wright with contributions from dept members – 4 issues: Jan, Feb, Mar, Apr	2	This looks embarrassingly silly today
117	Letter from C L (Chuck) Baker at Rand Corp to William Heising, 4/22/58 about "fatal error stops" in Fortran & how to override them	2	
118	*Proposed additions and modifications in the specifications for the Fortran system* 2/10/55 this note was retrieved by Jean Sammet from her files, probably around 6/78 for HOPL, see item # 17	2	
119	Letter from John McCarthy enclosing a listing for a machine language program FUNC, and chapters 1 & 2 describing a new compiler OMNITRAN	2	Includes the listing and the 2 chapters

Page 7 of 13

… and from sources like these, oral histories, etc., we can get a good impression of the course of the project.

Programmer's
Reference Manual
October 15, 1956

THE FORTRAN
AUTOMATIC CODING SYSTEM
FOR THE
IBM 704 EDPM®

This manual supersedes all earlier information about the FORTRAN system. It describes the system which will be made available during late 1956, and is intended to permit planning and FORTRAN coding in advance of that time. An Introductory Programmer's Manual and an Operator's Manual will also be issued.

APPLIED SCIENCE DIVISION
AND PROGRAMMING RESEARCH DEPT.
International Business Machines Corporation
590 Madison Ave., New York 22, N. Y.

WORKING COMMITTEE

J. W. BACKUS
R. J. BEEBER
S. BEST
R. GOLDBERG
H. L. HERRICK
R. A. HUGHES
University of California Radiation Laboratory, Livermore, Calif.
L. B. MITCHELL
R. A. NELSON
R. NUTT
United Aircraft Corp., East Hartford, Conn.
D. SAYRE
P. B. SHERIDAN
H. STERN
I. ZILLER

After what seemed like endless delays, the team prepared for their first release. The date, October 15, 1956, proved to be a bit premature.

Formal Rules for Forming Expressions. By repeated use of the following rules, all permissible expressions may be derived.

1. Any fixed point (floating point) constant, variable, or subscripted variable is an expression of the same mode. Thus 3 and I are fixed point expressions, and ALPHA and A(I,J,K) are floating point expressions.

2. If SOMEF is some function of n variables, and if E, F, ... , H are a set of n expressions of the correct modes for SOMEF, then SOMEF (E, F, ... , H) is an expression of the same mode as SOMEF.

3. If E is an expression, and if its first character is not + or −, then +E and −E are expressions of the same mode as E. Thus −A is an expression, but +−A is not.

4. If E is an expression, then (E) is an expression of the same mode as E. Thus (A), ((A)), (((A))), etc. are expressions.

5. If E and F are expressions of the same mode, and if the first character of F is not + or −, then

E + F
E − F
E * F
E / F

are expressions of the same mode. Thus A−+B and A/+B are not expressions. The characters +, −, *, and / denote addition, subtraction, multiplication, and division.

While computer scientists made fun of FORTRAN for the next few decades, it was actually a very powerful language with a compact and elegant reference manual, whose notation foreshadows the BNF Backus came up with later for the Algol project.

I. Ziller

Preliminary

Operator's Manual

April 8, 1957

THE FORTRAN

AUTOMATIC CODING SYSTEM

FOR THE

IBM 704 EDPM

This manual describes the use of FORTRAN 4-1-4-1.

Programming Research Department

International Business Machines Corporation

590 Madison Ave., New York 22, New York

Lest you think FORTRAN was “just” a compiler, remember there were no operating systems at that time.

RUNNING THE OBJECT PROGRAM

The binary deck which is produced when switch 1 is UP consists of the object program in relocatable binary, together with the four-card FORTRAN relocating loader UA CSB3 and appropriate control card and transfer card. The binary deck is thus ready for immediate loading and execution. For further details see the forthcoming SHARE write-up for UA CSB3.

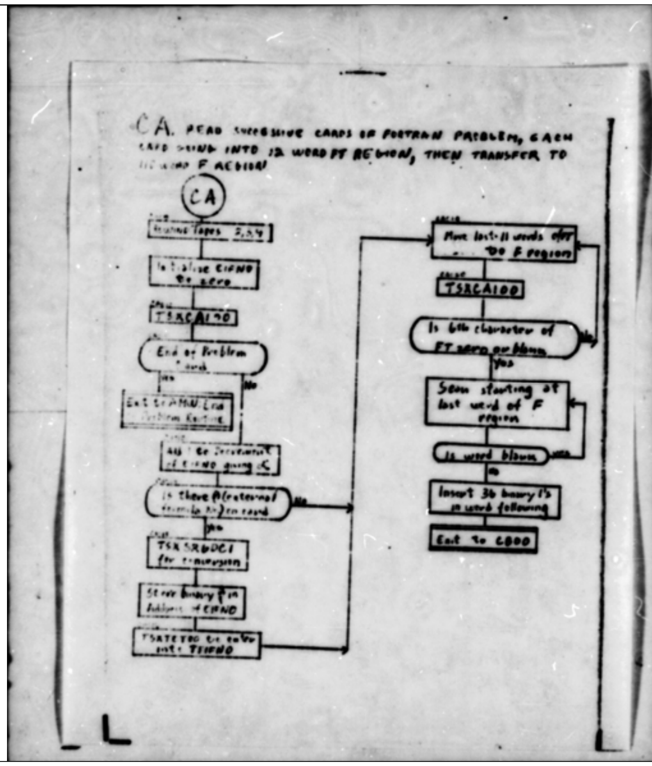
Details about using the binary tape form of the object program will be announced later.

The printer board to be used with FORTRAN object program is SHARE #2.

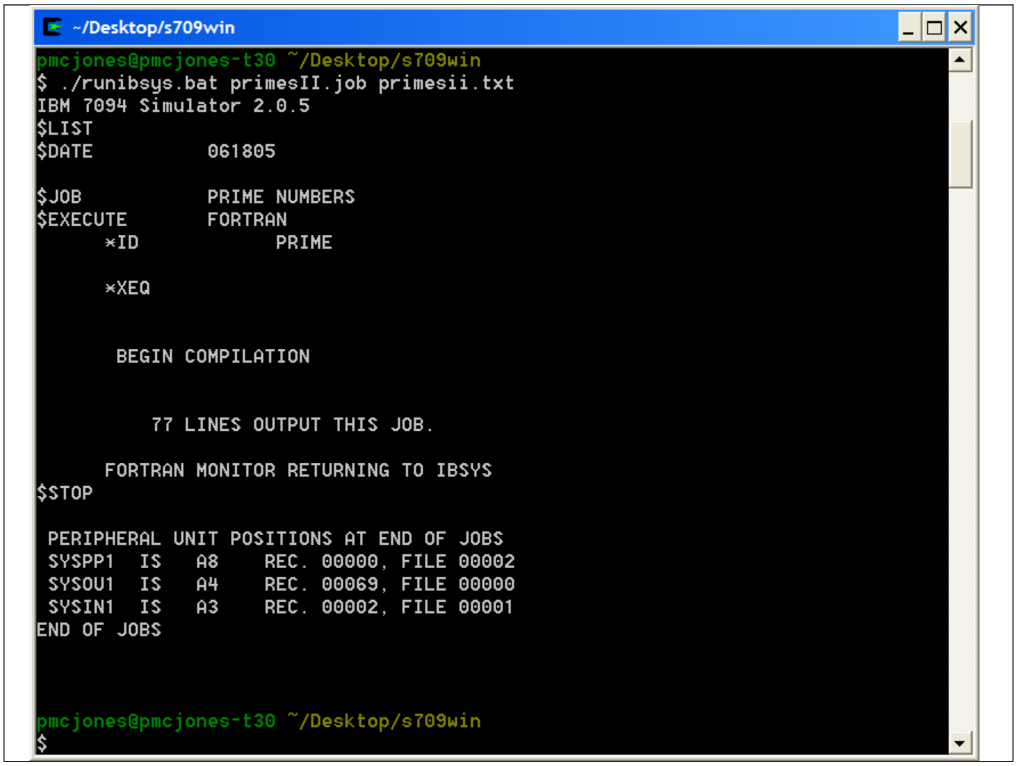
FORTRAN supplied a complete operating environment, with I/O, a loader, etc.

```
        REM  PASS1 / CALLS ON RDCIT1 AND MIH                                F4A01200
        REM       THIS ROUTINE SCANS THE COMPILED INSTRUCTIONS AND          F4A01210
        REM  DIVIDES THE OBJECT PROGRAM INTO BASIC BLOCKS.  A BASIC BLOCKF4A01220
        REM  IS A STRETCH OF PROGRAM WITH ONLY ONE ENTRY POINT AND ONE      F4A01230
        REM  EXIT POINT.  THE OUTPUT OF THIS ROUTINE IS BBLIST, A TABLE     F4A01240
        REM  LISTING THE BEGINNING OF EACH BASIC BLOCK (THE LOCATION        F4A01250
        REM  SYMBOL OF THE 1ST INSTRUCTION OF EACH BASIC BLOCK).  A TABLEF4A01260
        REM  CALLED DOLIST IS ALSO COMPILED OF THE BEGINNING AND END OF     F4A01270
        REM  EVERY LOOP IN THE OBJECT PROGRAM.                              F4A01280
        REM       PASS1 ALSO STARTS READING TIFGO, TRAD AND FRET FROM       F4A01290
        REM  TAPE FOR LATER PROCESSING.                                     F4A01300
PASS1   ZAC                                                                 F4A01310
        DCT                    BE SURE DIV-CHECK IND IS OFF.                F4A01320
        NOP                                                                 F4A01330
        SXD     CITCNT,0       CLEAR DECREMENT OF CIT COUNT.                F4A01340
        LDQ     CITCNT         DIVIDE NO. OF WORDS IN CIT BY                F4A01350
        STQ     ILNGTH         LENGTH OF RESERVED AREA TO                   F4A01360
        DVP     CTSPC1         FIND TOP OF LAST RECORD READ.                F4A01370
        DCT                                                                 F4A01380
        TSX     ERRM4,IR4      DIVIDE ERROR GO TO DIAGNOSTIC.               F4A01390
        REM                          LENGTH OF CITS PLACED IN CITCNT        F4A01400
        REM                       BY SEC 3 , LENGTH OF CIT AREA DEFINED F4A01410
        REM                          BY ASSEMBLY.                           F4A01420
        ADD     CTRD1          ADD BOTTOM OF AREA TO GET LAST LOADED        F4A01430
        STA     LAST1          ADDRESS+1 AND SET FOR EXIT ROUTINE.          F4A01440
        TSX     RDTBLS,IR2     START READING TIFGO.                         F4A01450
        TSX     (TAPE),IR4     READ FIRST RECORD OF CIT.                    F4A01460
                CTRD1,,(RBEC)                                               F4A01470
                CTLBL,,CTAPE                                                F4A01480
        LXA     (SCHU)+CTAPE,IR1                                            F4A01490
        LXA     CTRD1,IR2                                                   F4A01500
```

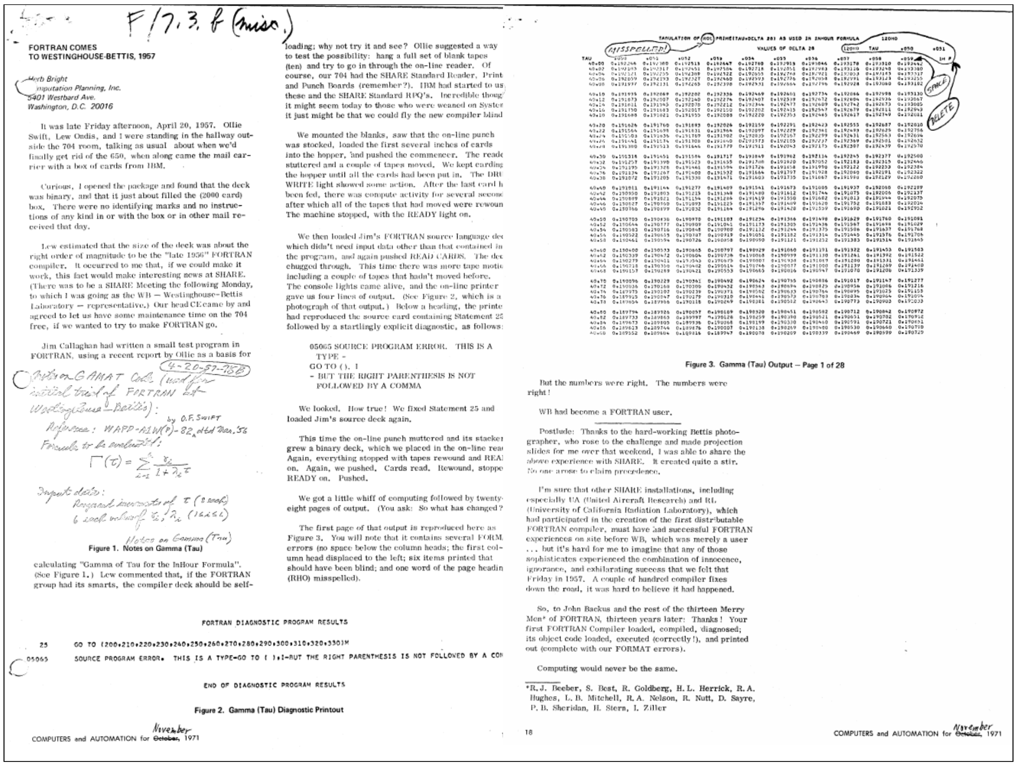
Now we’ve been harping on the idea that collecting and preserving the history of software means source code, so here’s a look at some. This is from Section Four, which produces a flow graph. Here is where the idea of “basic block” originated.



One of the artifacts that I've been chasing for a long time was referred to by Backus in his history paper as "The Tome". I also learned that Roy Nutt, who was on loan from UAC, saved microfilm from the FORTRAN project. Here's an image, from a microfiche borrowed from his son Micah.



Luckily, in addition to the microfiche, I learned that machine-readable source and object code was available, on the web site of Paul Pierce, a private collector in Portland, Oregon. An emulator begun by Pierce and further developed by Dave Pitts is able to run the FORTRAN II compiler, although we can't yet run the generated code.



F/7.3.f (misc.)

FORTRAN COMES
TO WESTINGHOUSE-BETTIS, 1957

Herb Bright
Computation Planning, Inc.
5401 Westbard Ave.
Washington, D.C. 20016

It was late Friday afternoon, April 20, 1957. Ollie Swift, Lew Ondis, and I were standing in the hallway outside the 704 room, talking as usual about when we'd finally get rid of the 650, when along came the mail carrier with a box of cards from IBM.

Curious, I opened the package and found that the deck was binary, and that it just about filled the (2000 card) box. There were no identifying marks and no instructions of any kind in or with the box or in other mail received that day.

Lew estimated that the size of the deck was about the right order of magnitude to be the "late 1956" FORTRAN compiler. It occurred to me that, if we could make it work, this fact would make interesting news at SHARE. (There was to be a SHARE Meeting the following Monday, to which I was going as the WB — Westinghouse-Bettis Laboratory — representative.) Our head CE came by and agreed to let us have some maintenance time on the 704 free, if we wanted to try to make FORTRAN go.

Jim Callaghan had written a small test program in FORTRAN, using a recent report by Ollie as a basis for

4-20-57-78B
Notes on GAMAT Code (used for initial trial of FORTRAN at Westinghouse-Bettis):
by O.F. SWIFT
Reference: WAPD-A1W(P)-82, dtd Mar. '56
Formula to be evaluated:

$$\Gamma(\tau) = \sum_{i=1}^{6} \frac{\gamma_i}{1+\lambda_i \tau}$$

Input data:
Required increments of τ (2 each)
6 each values of γ_i, λ_i $(1 \le i \le 6)$

Notes on Gamma (Tau)

Figure 1. Notes on Gamma (Tau)

calculating "Gamma of Tau for the InHour Formula". (See Figure 1.) Lew commented that, if the FORTRAN group had its smarts, the compiler deck should be self-loading; why not try it and see? Ollie suggested a way to test the possibility: hang a full set of blank tapes (ten) and try to go in through the on-line reader. Of course, our 704 had the SHARE Standard Reader, Print and Punch Boards (remember?). IBM had started to us these and the SHARE Standard RPQ's. Incredible thoug it might seem today to those who were weaned on Syste it just might be that we could fly the new compiler blind

We mounted the blanks, saw that the on-line punch was stocked, loaded the first several inches of cards into the hopper, and pushed the commencer. The reade stuttered and a couple of tapes moved. We kept carding the hopper until all the cards had been put in. The DRU WRITE light showed some action. After the last card h been fed, there was compute activity for several second after which all of the tapes that had moved were rewoun The machine stopped, with the READY light on.

We then loaded Jim's FORTRAN source language dec which didn't need input data other than that contained in the program, and again pushed READ CARDS. The dec chugged through. This time there was more tape motio including a couple of tapes that hadn't moved before. The console lights came alive, and the on-line printer gave us four lines of output. (See Figure 2, which is a photograph of that output.) Below a heading, the printe had reproduced the source card containing Statement 25 followed by a startlingly explicit diagnostic, as follows:

```
05065 SOURCE PROGRAM ERROR.  THIS IS A
  TYPE -
GO TO (), I
- BUT THE RIGHT PARENTHESIS IS NOT
  FOLLOWED BY A COMMA
```

We looked. How true! We fixed Statement 25 and loaded Jim's source deck again.

This time the on-line punch muttered and its stacke grew a binary deck, which we placed in the on-line rea Again, everything stopped with tapes rewound and REA on. Again, we pushed. Cards read. Rewound, stoppe READY on. Pushed.

We got a little whiff of computing followed by twenty-eight pages of output. (You ask: So what has changed?

The first page of that output is reproduced here as Figure 3. You will note that it contains several FORM errors (no space below the column heads; the first column head displaced to the left; six items printed that should have been blind; and one word of the page headin (RHO) misspelled).

```
                FORTRAN DIAGNOSTIC PROGRAM RESULTS

25      GO TO (200,210,220,230,240,250,260,270,280,290,300,310,320,330)M

05065   SOURCE PROGRAM ERROR.  THIS IS A TYPE-GO TO ( ),I-BUT THE RIGHT PARENTHESIS IS NOT FOLLOWED BY A CO

                END OF DIAGNOSTIC PROGRAM RESULTS
```

Figure 2. Gamma (Tau) Diagnostic Printout

TABULATION OF (RHO) PROHETTAU-DELTA 20) AS USED IN INHOUR FORMULA

VALUES OF DELTA 28

Figure 3. Gamma (Tau) Output — Page 1 of 28

But the numbers were right. The numbers were right!

WB had become a FORTRAN user.

Postlude: Thanks to the hard-working Bettis photographer, who rose to the challenge and made projection slides for me over that weekend, I was able to share the above experience with SHARE. It created quite a stir. No one arose to claim precedence.

I'm sure that other SHARE installations, including especially UA (United Aircraft Research) and RL (University of California Radiation Laboratory), which had participated in the creation of the first distributable FORTRAN compiler, must have had successful FORTRAN experiences on site before WB, which was merely a user . . . but it's hard for me to imagine that any of those sophisticates experienced the combination of innocence, ignorance, and exhilarating success that we felt that Friday in 1957. A couple of hundred compiler fixes down the road, it was hard to believe it had happened.

So, to John Backus and the rest of the thirteen Merry Men* of FORTRAN, thirteen years later: Thanks! Your first FORTRAN Compiler loaded, compiled, diagnosed; its object code loaded, executed (correctly!), and printed out (complete with our FORMAT errors).

Computing would never be the same.

*R. J. Beeber, S. Best, R. Goldberg, H. L. Herrick, R. A. Hughes, L. B. Mitchell, R. A. Nelson, R. Nutt, D. Sayre, P. B. Sheridan, H. Stern, I. Ziller

FORTRAN shipped in April 1957, and the users jumped on it immediately – as Herb Bright reported in 1971, they were able to get it running after receiving a binary deck unaccompanied by documentation.

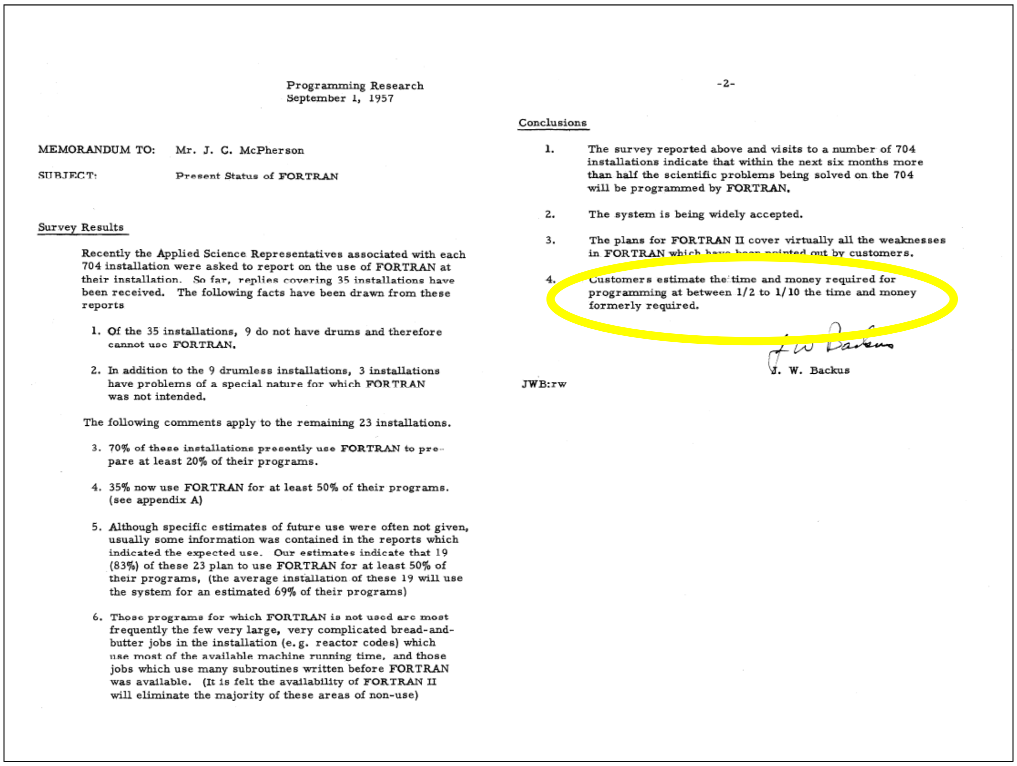
August 28, 1957

Preliminary Report: Proposed Specifications for FORTRAN II for the 704

General

FORTRAN has now had some use at a large number of installations and has been used heavily for some time at at least six installations. A large number plan to use the system for the majority of their programming work. Difficulties in using FORTRAN have fallen into two categories (1) Difficulties arising in the initial period of putting the system into use, and (2) Difficulties arising from the properties of the system. Problems in the first category are being solved by on-the-spot expert assistance to those installations experiencing an undue amount of difficulty. The consensus of opinion concerning difficulties in category (2) is that the principal areas needing improvement are (a) facilities for debugging source programs and (b) facilities for creating and using subroutines. FORTRAN II will contain improved facilities in both these areas along the lines described below.

By August, Backus's group had helped put out a lot of fires and was planning the next version, with improved debugging and the ability to define and use separately-compiled subroutines.



Programming Research
September 1, 1957

MEMORANDUM TO: Mr. J. C. McPherson

SUBJECT: Present Status of FORTRAN

Survey Results

Recently the Applied Science Representatives associated with each 704 installation were asked to report on the use of FORTRAN at their installation. So far, replies covering 35 installations have been received. The following facts have been drawn from these reports

1. Of the 35 installations, 9 do not have drums and therefore cannot use FORTRAN.

2. In addition to the 9 drumless installations, 3 installations have problems of a special nature for which FORTRAN was not intended.

The following comments apply to the remaining 23 installations.

3. 70% of these installations presently use FORTRAN to prepare at least 20% of their programs.

4. 35% now use FORTRAN for at least 50% of their programs. (see appendix A)

5. Although specific estimates of future use were often not given, usually some information was contained in the reports which indicated the expected use. Our estimates indicate that 19 (83%) of these 23 plan to use FORTRAN for at least 50% of their programs, (the average installation of these 19 will use the system for an estimated 69% of their programs)

6. Those programs for which FORTRAN is not used are most frequently the few very large, very complicated bread-and-butter jobs in the installation (e.g. reactor codes) which use most of the available machine running time, and those jobs which use many subroutines written before FORTRAN was available. (It is felt the availability of FORTRAN II will eliminate the majority of these areas of non-use)

-2-

Conclusions

1. The survey reported above and visits to a number of 704 installations indicate that within the next six months more than half the scientific problems being solved on the 704 will be programmed by FORTRAN.

2. The system is being widely accepted.

3. The plans for FORTRAN II cover virtually all the weaknesses in FORTRAN which have been pointed out by customers.

4. Customers estimate the time and money required for programming at between 1/2 to 1/10 the time and money formerly required.

J W Backus

J. W. Backus

JWB:rw

They surveyed the users, and found rapid adoption and based on estimates of large gains in programmer productivity.

Programming
Research

Mr. J. W. Backus

WHQ

September 23, 1957

In connection with the memorandum on proposed specifications for Fortran II for the 704, you will recall my request for a statement of what the proposed change will produce in the way of added flexibility. This would include an indication of the features proposed for Fortran for the 709 and Comtran which will be realized in Fortran II for the 704.

JCMcP:im

John C. McPherson

However the fifteen or twenty man-years that had been invested in FORTRAN were without precedent for a software project, and IBM management was a bit anxious and wanted to make sure that the software for the upcoming IBM 709 was suitably advanced.

Programming Research
November 5, 1957

MEMORANDUM TO: Mr. J. C. McPherson

SUBJECT: FORTRAN II

REFERENCE: Your memo of 9/23/57

The added flexibility of FORTRAN II stems from the ability to add virtually any procedure as a new statement in the language. It is therefore difficult to summarize the features which will be added. Some examples undoubtedly are: matrix, complex and double precision arithmetic; monitor routines for manipulating the arrangement of subroutines in the store; routines for handling BCD information; special input-output operations.

With the exception of some format and spelling changes proposed for FORTRAN for the 709, FORTRAN II specs go considerably beyond any previous specs for the 709 system, in ways which we have discussed (independent subroutines; substitution of argument variables in calling statement for dummy variables in subroutine).

J W Backus

J. W. Backus

JWB:rw

Backus confidently predicted that the planned FORTRAN II extensions would be well received.

Dennis E. Hamilton. Impact of FORTRAN II language changes.
Personal communication to Paul McJones, April 2005

"However, the impact of small changes and improvements can be immense. The ability to build Fortran programs out of independently-compilable modules and to have the ability to decompose into functions and subroutines using Fortran or any other tool that produced compatible code (usually the assembler, in those days) had an immense impact. In Fortran I programs were one giant file and there was no modularization structure. That small change in Fortran II was earthshaking in terms of software development and, I think, the endurance of Fortran as a technical-software programming tool.

It also changed the way that computers had to operate to make software building and use work more smoothly. I think it is no coincidence that this paralleled increased interest in operating systems (called things like tape monitors, at the time) and the use of the computer for organizing the data processing workflows. (There was also a lot of resistance to operating systems in those days.)"

Via the Internet, I ran across someone whose memories of the advent of FORTRAN II supported that prediction.

April, 1960

SYSTEMS MANUAL

FOR 704 FORTRAN

AND 709 FORTRAN

Applied Programming Department
International Business Machines Corporatior
590 Madison Avenue
New York 22, New York

As the importance of FORTRAN became clear, more and more people began to work on. One of the benefits was improved documentation. This was before IBM unbundled its software, so interested customers were able to order copies of the source code and documentation.

VII

SECTION FOUR

PART 1

The first task of this part is to divide the object program into basic blocks, a basic block being a stretch of program with but one entry point, and one exit point. In order to do this, a pass is made over CIT looking for transfers, tests and skip type instructions. Transfer and conditional transfer addresses, and the locations of instructions following skip type instructions or TXLs (end tests of Dos), are all entered in the BBLIST table once only, in algebraic order, by means of a binary search technique. The assigned Go To instructions are ignored for the moment.

During this pass, when a TXL is encountered, both its location and address are entered in the DOLIST table, thus providing a list of the beginning and end locations of all Dos, in end location order.

This is the chapter that describes the source code I showed you earlier. At the Smithsonian, I found a listing of FORTRAN II from around this same time period that IBM had sent to Peter Z. Ingerman, then at the University of Pennsylvania.

PROGRAMMING LANGUAGES AND THEIR COMPILERS

Preliminary Notes

Second Revised Version, April 1970

John Cocke and J. T. Schwartz

Courant Institute of Mathematical Sciences
New York University

C. Index register allocation for Algebraic languages

Whereas rigorously based results applying to simplified models of register allocation in linearly ordered code are available (cf. Horwitz, Karp, Miller, and Winograd [1]), no such results have been derived for code containing even simple loops, much less for code with general flow. Thus register allocation in the presence of flow must use a heuristic lacking complete theoretical foundation. The particular method used will depend strongly on the extent to which global information is to be collected in support of register allocation. Machine structure will, of course, play a fundamental roll as well. The simplest scheme, incorporated in many industrial compilers, is to put a newly generated quantity in an empty register if possible; if none is available, the contents of the register used farthest away in the following code is displaced. In this method, register contents are remembered over the extent of a basic block, but all registers are stored and reloaded at each label. This method requires only local look-ahead.

A more elaborate register allocation heuristic has been outlined in a preceding section. Here we shall describe the allocation methods actually used in two interesting production compilers, the original FORTRAN I compiler and a FORTRAN compiler for the RCA SPECTRA 70.

The method which was used for index register allocation in the FORTRAN I compiler is described in Backus et al [1];

-510-

Because of the ACS project at IBM, John Cocke became very interested in compiler technology. Fifteen years after the FORTRAN compiler was written, he and Jack Schwartz found the FORTRAN index register allocation algorithms still worthy of inclusion in their classic unpublished book.

Here's an picture of John Backus and John Cocke in Atlantic City, New Jersey, date unknown.

I hope you've enjoyed your tour of the FORTRAN exhibit of the Computer History Museum.

To take your own tour

- http://community.computerhistory.org/scc/projects/FORTRAN/
- http://community.computerhistory.org/scc/projects/LISP/
- And my blog with the “behind the scenes” story:
  - http://www.mcjones.org/dustydecks/