# IBM-650 simulator

World’s first mass-produced computer

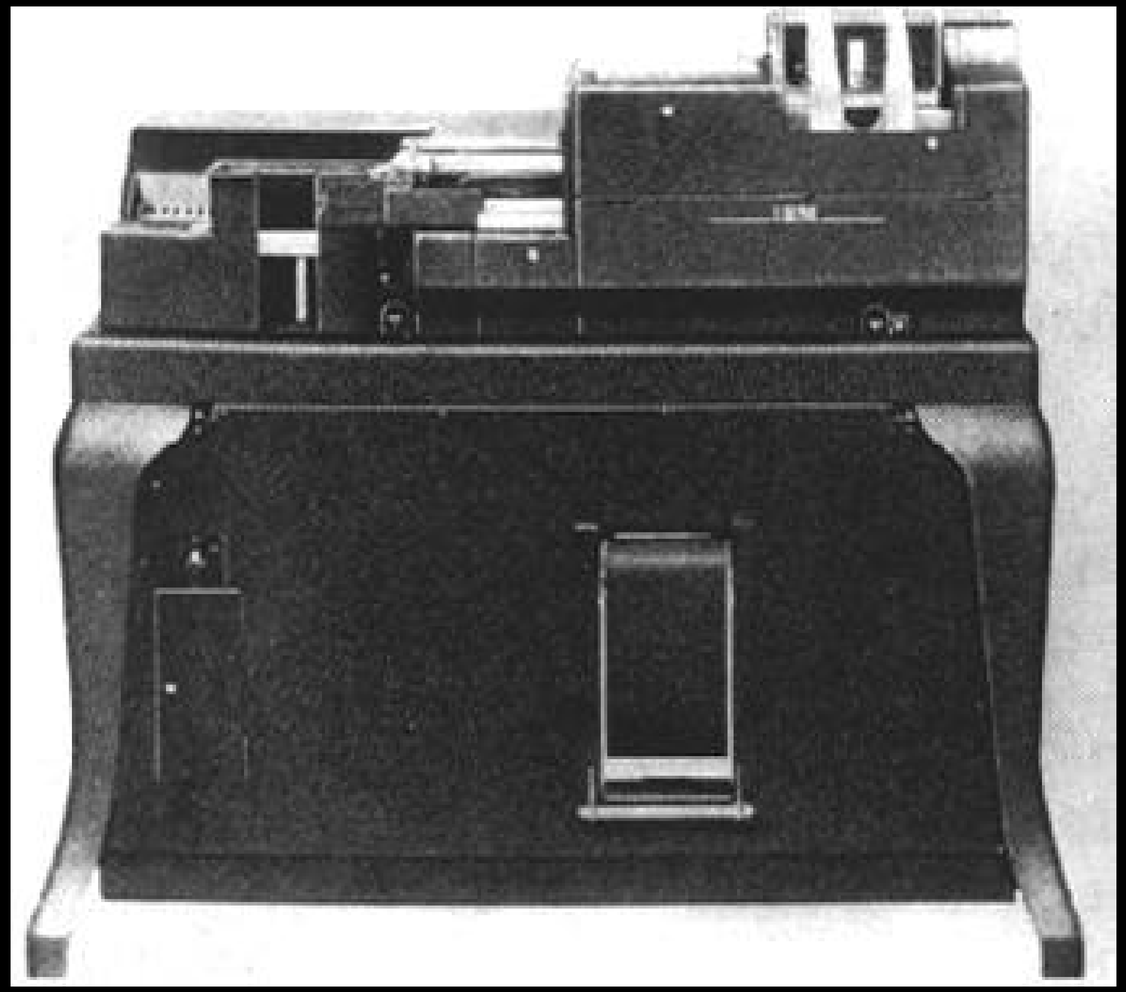
Brief history:

# Background

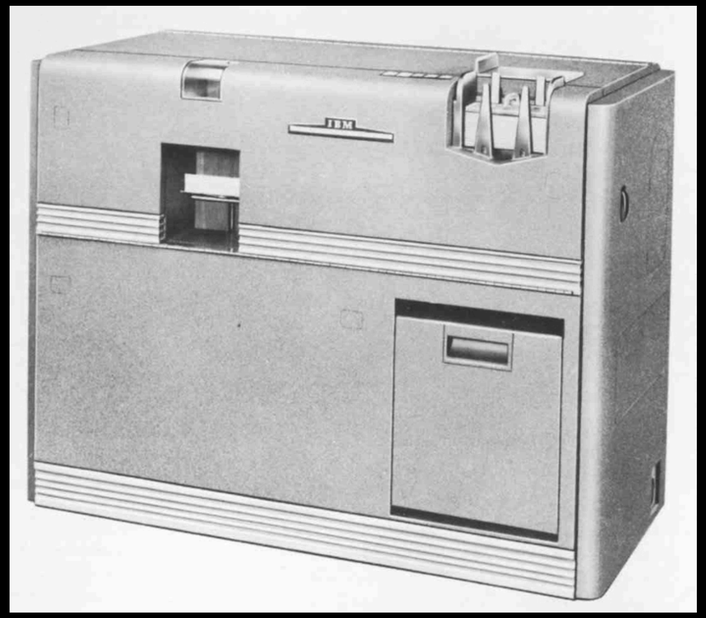
As computing technology trickled down to the masses, automating calculations became an attractive option for big business.

# IBM 601

Introduced in 1931, used for basic record keeping. First IBM machine capable of multiplication.

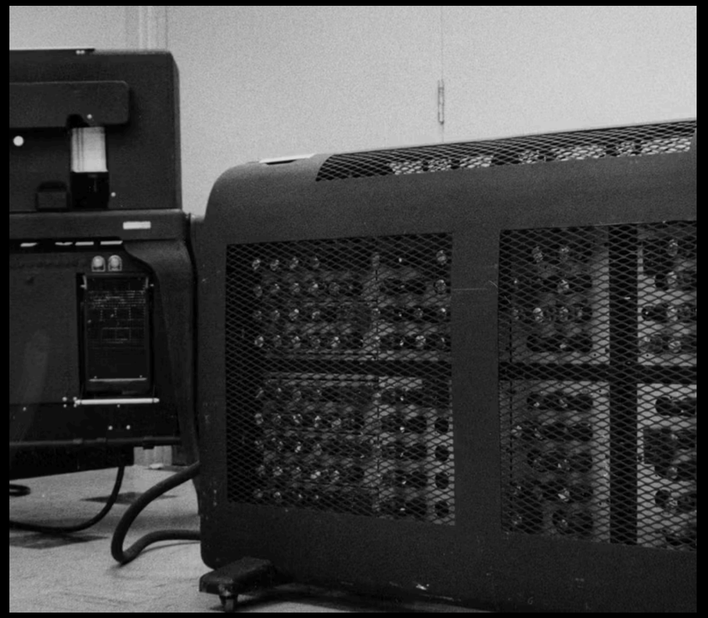


# IBM 6-series: 602-604




## 602

Advanced, smaller version of 601. Introduced more arithmetic operations.



## 603

Released alongside 602, fully electronic. Demand for this machine surprised IBM.

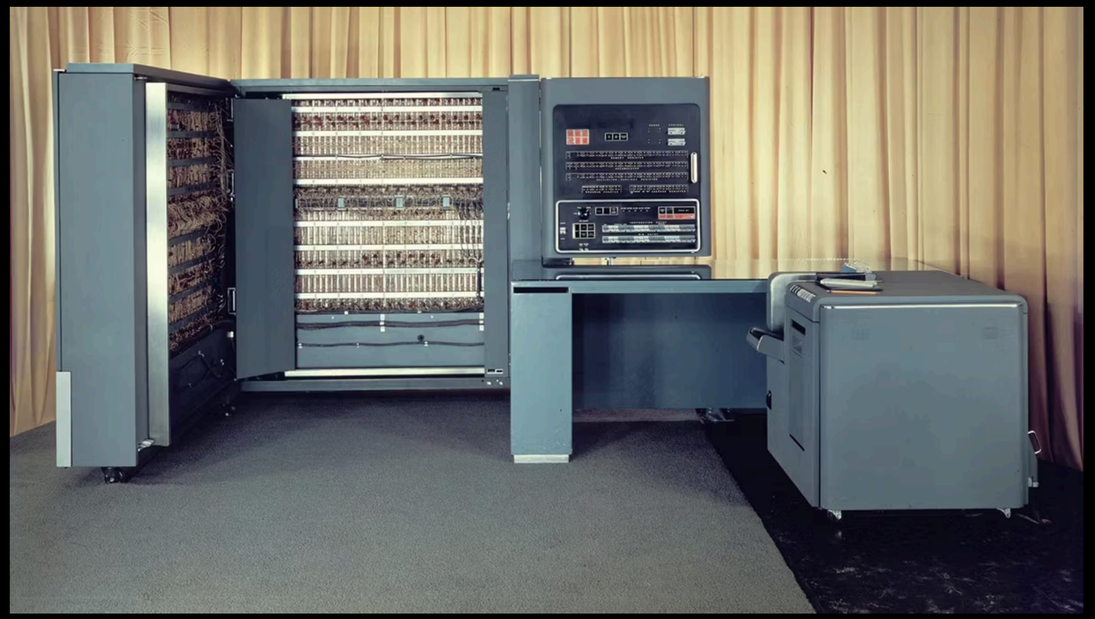
## 604

Improved 603. Now a component of the larger CPC and CPC-II systems

# The 650 and 701/2

## 650

Cheaper, slower. Magnetic drum memory, decimal words. Designed to be (relatively) affordable and easy to use. 2000+ units made



## 701 and 702

Very expensive to buy and operate. Tube memory, binary words. Top of the line at its time, very influential. 19 units made.

# Specs

## Processor

125 KHz, Digit-by-Digit serial

## Memory

1000-4000 10-digit words on a magnetic drum. 2.5ms average access time

## I/O

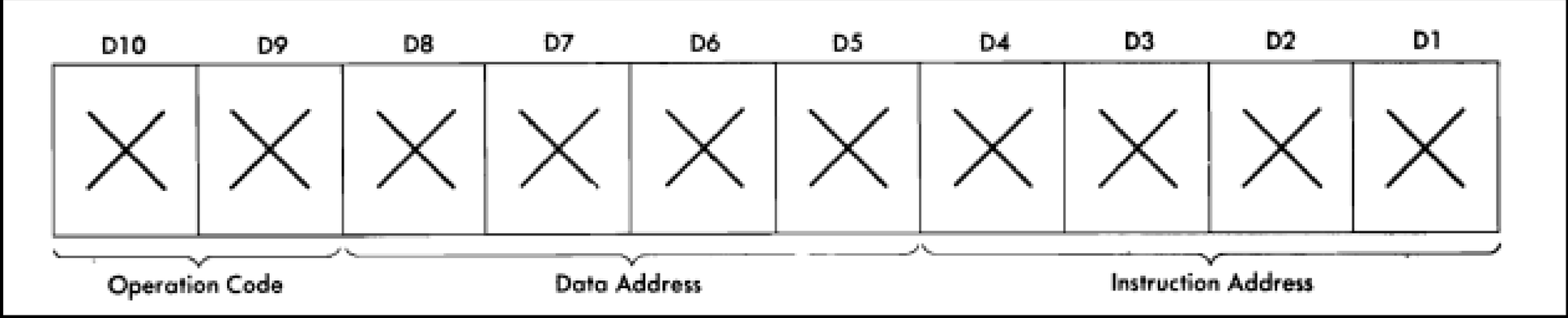
Separate Punch/Read unit, 80-column cards

## ISA

100 opcodes (44 for the base machine)
20-digit accumulator

## Word structure

Each instruction contains an opcode, the address for data, and the address of the next instruction. Explicit ordering of instructions allows to optimize for the quirks of rotating drum memory. IBM user manuals provide optimization recommendations.



## Opcodes

Arithmetic is done via a split accumulator register.

| Code | Abbreviation | Operation |
|---|---|---|
| 00 | NO OP | No Operation |
| 01 | STOP | Stop |
| 10 | AU | Add to Upper |
| 11 | SU | Subtract from Upper |
| 14 | DIV | Divide |
| 15 | AL | Add to Lower |
| 16 | SL | Subtract from Lower |
| 17 | AABL | Add Absolute Value to Lower |
| 18 | SABL | Subtract Absolute Value from Lower |
| 19 | MULT | Multiply |
| 20 | STL | Store lower |
| 21 | STU | Store upper |
| 22 | STDA | Store Lower Data Address |
| 23 | STIA | Store Lower Instruction Address |
| 24 | STD | Store Distributor |
| 30 | SRT | Shift Right |
| 31 | SRD | Shift and Round |
| 35 | SLT | Shift Left |
| 36 | SCT | Shift Left and Count |
| 44 | BRNZU | Branch on Non-zero in Upper |
| 45 | BRNZ | Branch on Non-Zero |
| 46 | BR MIN | Branch on Minus |
| 47 | BR OV | Branch on Overflow |
| 60 | RAU | Reset-Add to Upper |
| 61 | RSU | Reset-Subtract from Upper |
| 64 | DIV RU | Divide-Reset Remainder |
| 65 | RAL | Reset-Add to Lower |
| 66 | RSL | Reset-Subtract from Lower |
| 67 | RAABL | Reset-Add Absolute Value to Lower |
| 68 | RSABL | Reset-Subtract Absolute Value from Lower |
| 69 | LD | Load Distributor |
| 70 | RD1 | Read into Input Storage Area 1 |
| 71 | WR1 | Write (Print or Punch) from Output Storage Area 1 |
| 72 | RC1 | Read Conditional into Input Storage Area 1 |
| 84 | TLU | Table Lookup |
| 90 | BRD 10 | Branch on 8 in 10th Position of Distributor |
| 91 | BRD 1 | Branch on 8 in 1st Position of Distributor |
| 92-99 | BRD 2-9 | Branch on 8 in 2nd through 9th Position of Distributor |

*Figure II-1. Operation Codes*

# Emulator demo

Open SimH - Bringing Antiquities back to life

| *Hardware Console Action* | *Equivalent Simulation Command* |
|---|---|
| Set PROGRAMMED switch to RUN \| STOP | DEPOSIT CSWPS 0 \| 1 |
| Set OVERFLOW switch to SENSE \| STOP | DEPOSIT CSWOS 0 \| 1 |
| Set HALF CYCLE switch to RUN \| HALF | DEPOSIT HALF 0 \| 1 |
| Set ERROR switch | Not simulated |
| Set CONTROL to ADDRESS STOP | BREAK <address> |
| Setting the Console Switches | DEPOSIT CSW <value> |
| Display Lower Accumulator | EXAMINE ACCLO |
| Display Upper Accumulator | EXAMINE ACCUP |
| Display Distributor | EXAMINE DIST |
| Display Program Register | EXAMINE PR |
| Display Read-Out Storage | EXAMINE <address> |
| Display Read-In Storage | DEPOSIT <address> <value> |
| Press TRANSFER Key | DEPOSIT AR <address> |
| Press PROGRAM START Key | GO |
| Press PROGRAM STOP Key | ^E on console |
| Press COMPUTER RESET Key | RESET |

```
sim> d 100 6001030102
sim> d 101 2
sim> d 103 3
sim> d 102 1001010104
sim> d 104 2101050106
sim> d 106 0100000107
sim> d 107 0000000000
sim> e 100-107
100:     6001030102+
101:     0000000002+
102:     1001010104+
103:     0000000003+
104:     2101050106+
105:     0000000000+
106:     0100000107+
107:     0000000000+
sim> d ar 100
sim> go

Programmed Stop, IC: 00106 ( 0100000107+   HLT   0000  0107 )
sim> e 105
105:     0000000005+
sim>
```

**Adding 2+3.**

(Note data words mixed in among instructions)

## IBM 650 software

### SOAP

**Assembly language**

Many flavors: SOAP-II, SOAP4, SuperSoap

### FOR TRANSIT

**High-level programming language**

Subset of FORTRAN, ported specifically to 650.
Compiles to SOAP via an intermediate representation.

### L1/L2

**Bell labs interpreter**

Early VM. Notably adds floating-point operations.

# SOAP-II features

- Instruction placement optimization
- ASM directives for code structuring
- Space for comments

# SOAP Demo

A deck of cards

## Card deck

- `soapII.dck` (1-word per card load cards)
- `soapII_condensed_card.dck` (7 words per card load cards)

Example.I

Prepare a table of

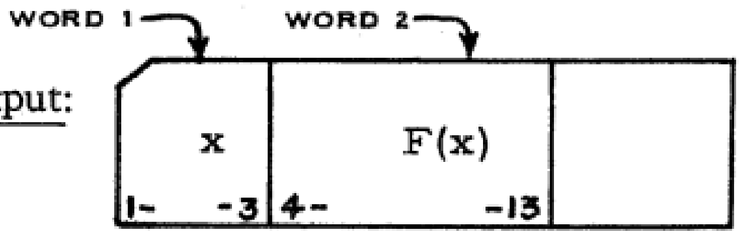
$$F(x) = Ax^2 + Bx + C$$

for

$$x = 1, 2, \ldots, 100$$

assuming A, B, C integers and $|F(x)| < 10^{10}$.

Output:



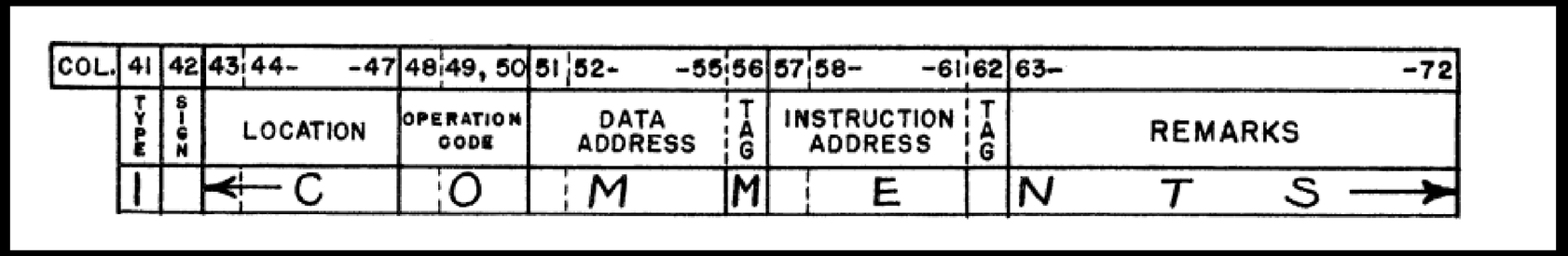
# 650 SŌAP II CODING FORM

| COL. | 41 | 42 | 43 44- -47 | 48 49, 50 | 51 52- -55 | 56 | 57 58- -61 | 62 | 63- -72 | ACCUMULATOR | | DISTRIBUTOR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TYPE | SIGN | LOCATION | OPERATION CODE | DATA ADDRESS | TAG | INSTRUCTION ADDRESS | TAG | REMARKS | UPPER 8003 | LOWER 8002 | 8001 |
| | 1 | | EXAMP | LE | 1 b CAL | C | U LATE | | F OF X | | | |
| | 1 | | | | | | | | | | | |
| | | | | BLR | 1951 | | 1960 | | READ AREA | | | |
| | | | | REG | P 0027 | | 0028 | | PUNCH AREA | | | |
| | 1 | | | | | | | | | | | |
| | | | SETX | RAU | ŌNE | | STX | | SET X | 1 | | |
| | | | STX | STU | P0001 | | | | TO 1 | X | | |
| | | | | MPY | A | | | | CALCULATE | | AX | |
| | | | | ALO | B | | | | F | | AX+B | |
| | | | | RAU | 8002 | | | | | AX+B | | |
| | | | | MPY | P0001 | | | | | | (AX+B)X | |
| | | | | ALO | C | | | | | | (AX+B)X+C | |
| | | | | STL | P0002 | | | | | | | |
| | | | | PCH | P0001 | | | | PUNCH | | | |
| | | | | RAU | P0001 | | | | IS X MAX | X | | |
| | | | | SUP | X MAX | | | | | X-100 | | |
| | | | | NZU | | | 9999 | | | | | |
| | | | | AUP | 101 | | STX | | STEP X | X+1 | | |
| | 1 | | | | | | | | | | | |
| | | | ŌNE | 00 | 0000 | | 0001 | | CONSTANTS | | | |
| | | | X MAX | 00 | 0000 | | 0100 | | | | | |
| | | | 101 | 00 | 0000 | | 0101 | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

# Future plans

Write a SOAP program of my own.
Experiment with FOR TRANSIT.
Advance in my proficiency with the emulator and the SIMH ecosystem.

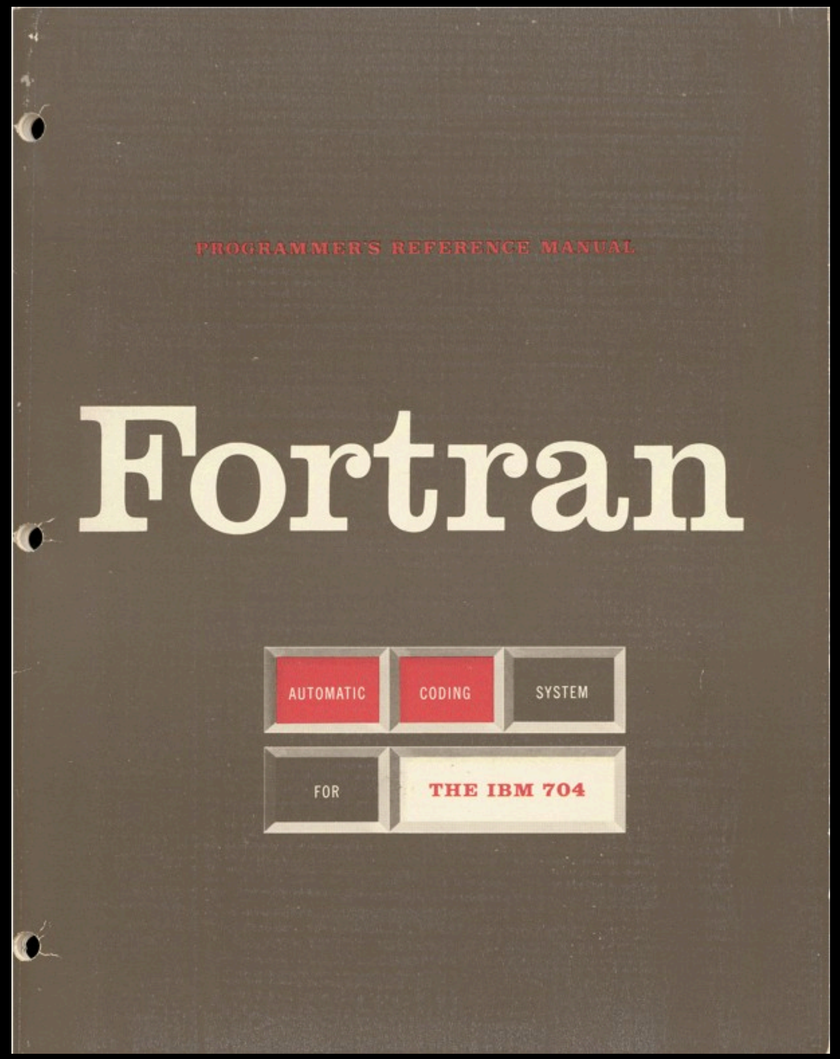
# Many thanks!



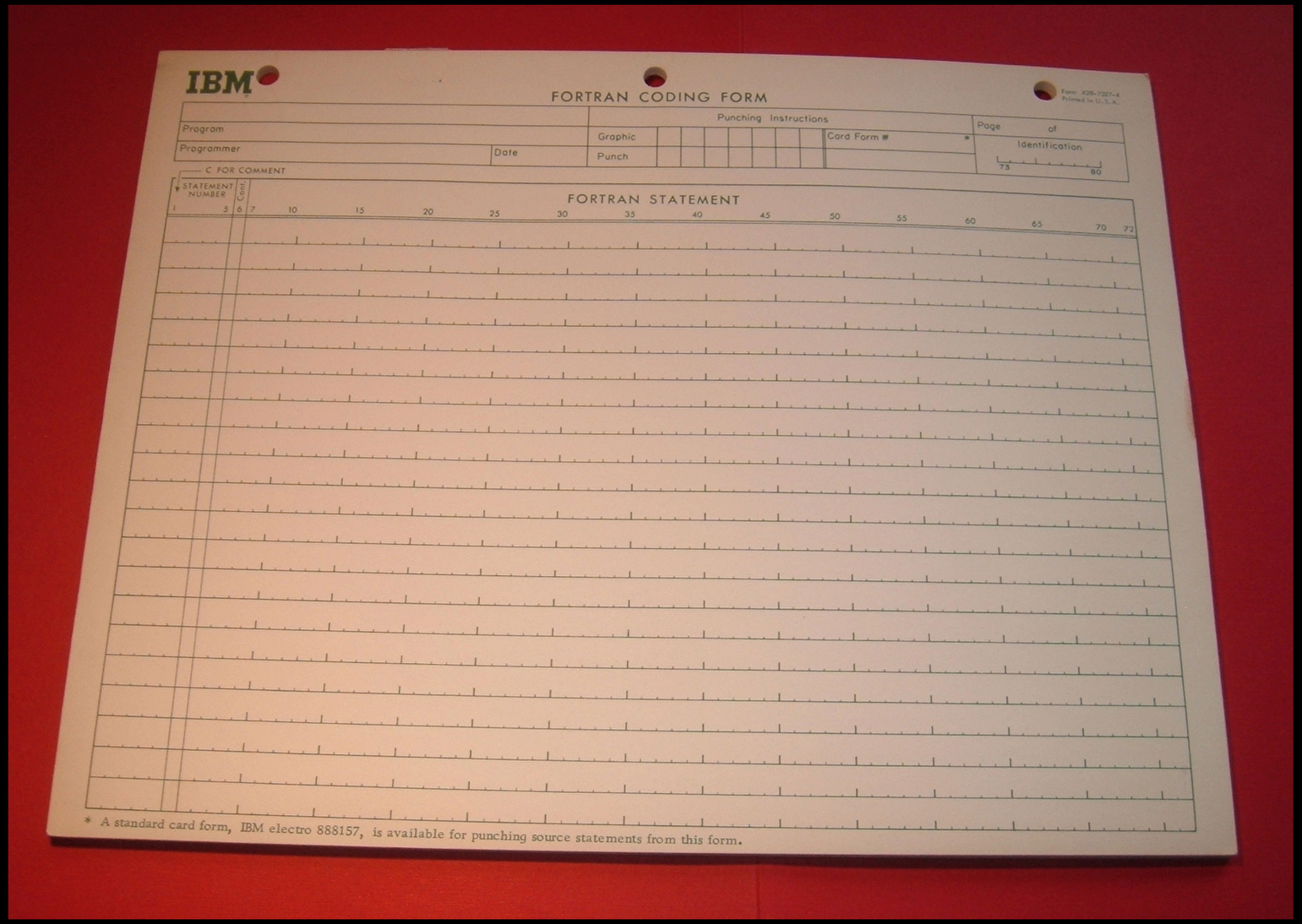
| COL. | 41 | 42 | 43 44- -47 | 48 49, 50 | 51 52- -55 | 56 | 57 58- -61 | 62 | 63- -72 |
|---|---|---|---|---|---|---|---|---|---|
| | TYPE | SIGN | LOCATION | OPERATION CODE | DATA ADDRESS | TAG | INSTRUCTION ADDRESS | TAG | REMARKS |
| | I | | ← C | O | M | M | E | | N T S → |

# Part 2

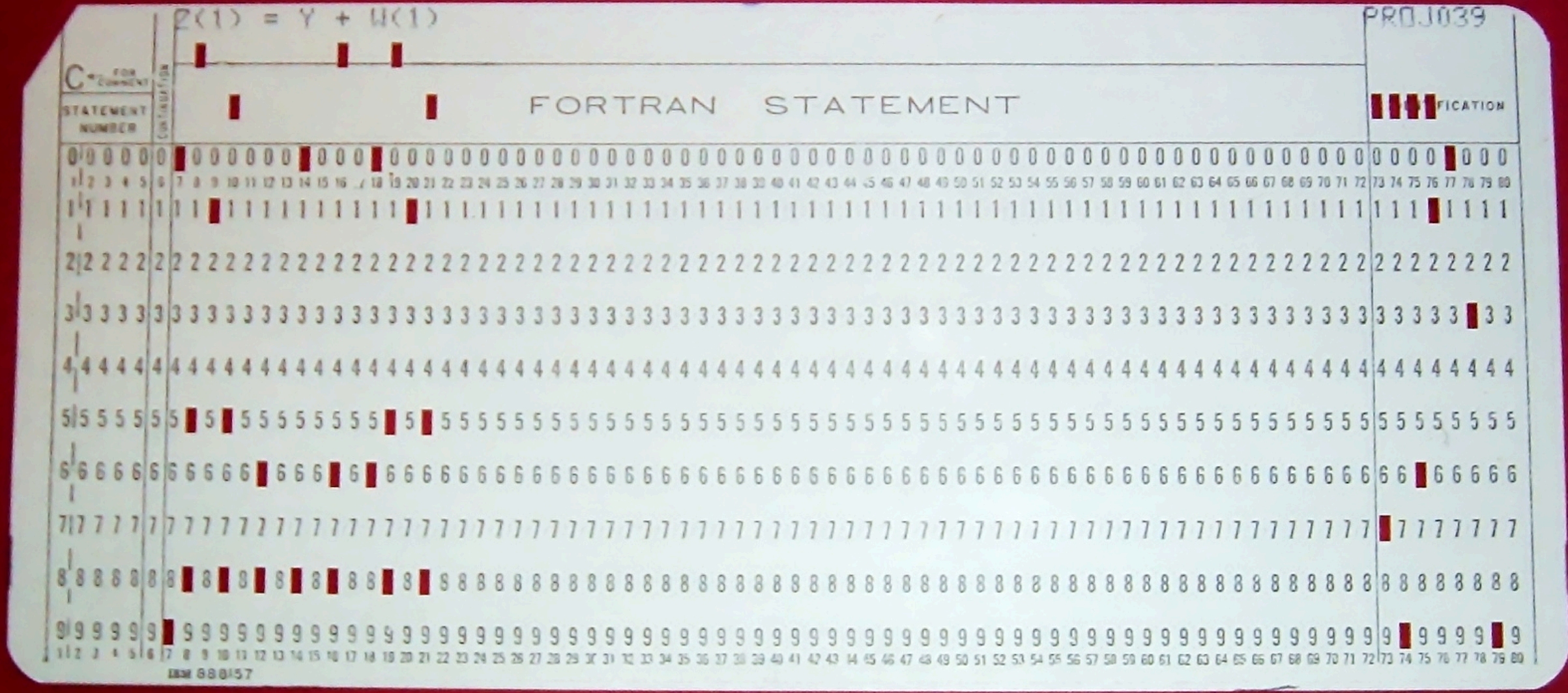
Exploring advanced IBM 650 programming




- Namesake: Formula Translation(-or)
- Idea: 1952-53.
- Designed to improve ease of programming over assembly.
- Delivered: 1957, for the IBM 701




FORTRAN coding form, to be punched onto cards




Single line to a punch card


Card punch machine

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT -
C INTEGER VARIABLES START WITH I,J,K,L,M OR N
      READ(5,501) IA,IB,IC
  501 FORMAT(3I5)
      IF (IA) 701, 777, 701
  701 IF (IB) 702, 777, 702
  702 IF (IC) 703, 777, 703
  777 STOP 1
  703 S = (IA + IB + IC) / 2.0
      AREA = SQRT( S * (S - IA) * (S - IB) * (S - IC) )
      WRITE(6,801) IA,IB,IC,AREA
  801 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
     $13H SQUARE UNITS)
      STOP
      END
```

Sample program (FORTRAN II)

# FOR TRANSIT

IBM’s answer to the demand for a high-level programming language for the IBM 650. FORTRAN with much smaller footprint.

# Disparities

5 max variable name length
Max 2 array dimensions (not 3)
No line printer = no formatting and literals
Very limited subroutines (functions)

# Namesake?

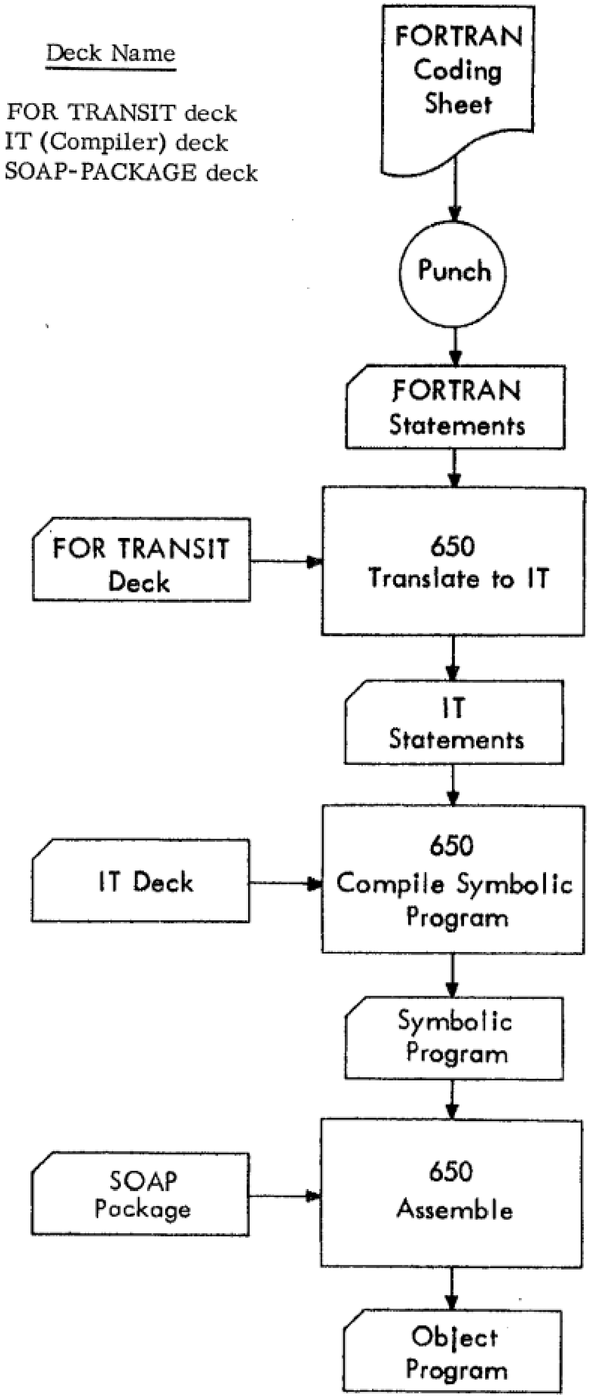
One or more of three meanings:

- FORTRAN-S(oap)-IT, describing the translation process
- FOR TRANSIT(ion), indicating that it was intended to ease upgrades to the 704
- FORTRAN's IT, in the sense of adding a FORTRAN front-end to the IT compiler

# Compilation

Compiles to IT (intermediate translation), then to SOAP, then to machine code

| Deck No. | Deck Name |
| --- | --- |
| 1 | FOR TRANSIT deck |
| 2 | IT (Compiler) deck |
| 3 | SOAP-PACKAGE deck |



| C ← FOR COMMENT<br>STATEMENT NUMBER (1–5) | CONTINUATION (6) | FORTRAN STATEMENT (7–72) |
|---|---|---|
| | | A = 3. |
| | | B = 1.7 |
| | | C = -31.92 |
| | | ROOT = (-B +SQRTF(B**2-4.*A*C))/(2.*A) |
| 11 | | PUNCH 1, ROOT |
| | | END |
| | | |

Sample program

| | |
|---|---|
| fortransit_example_1_src.txt | Sample Fortransit source. Prints prime numbers < 50 |
| fortransit_example_2_src.txt | Sample Fortransit source as in manual (rectangular |
| fortransit_example_2_data.txt | matrix multiplication) and input data |
| fortransit_example_3_src.txt | Sample Fortransit source as in manual |
| fortransit_example_4_src.txt | Sample Fortransit source. Package functions test |
| fortransit_example_5_src.txt | Sample Fortransit source. Card punch graphics |

**Included in SIMH**

# Let’s run some programs!