
ORACLE®

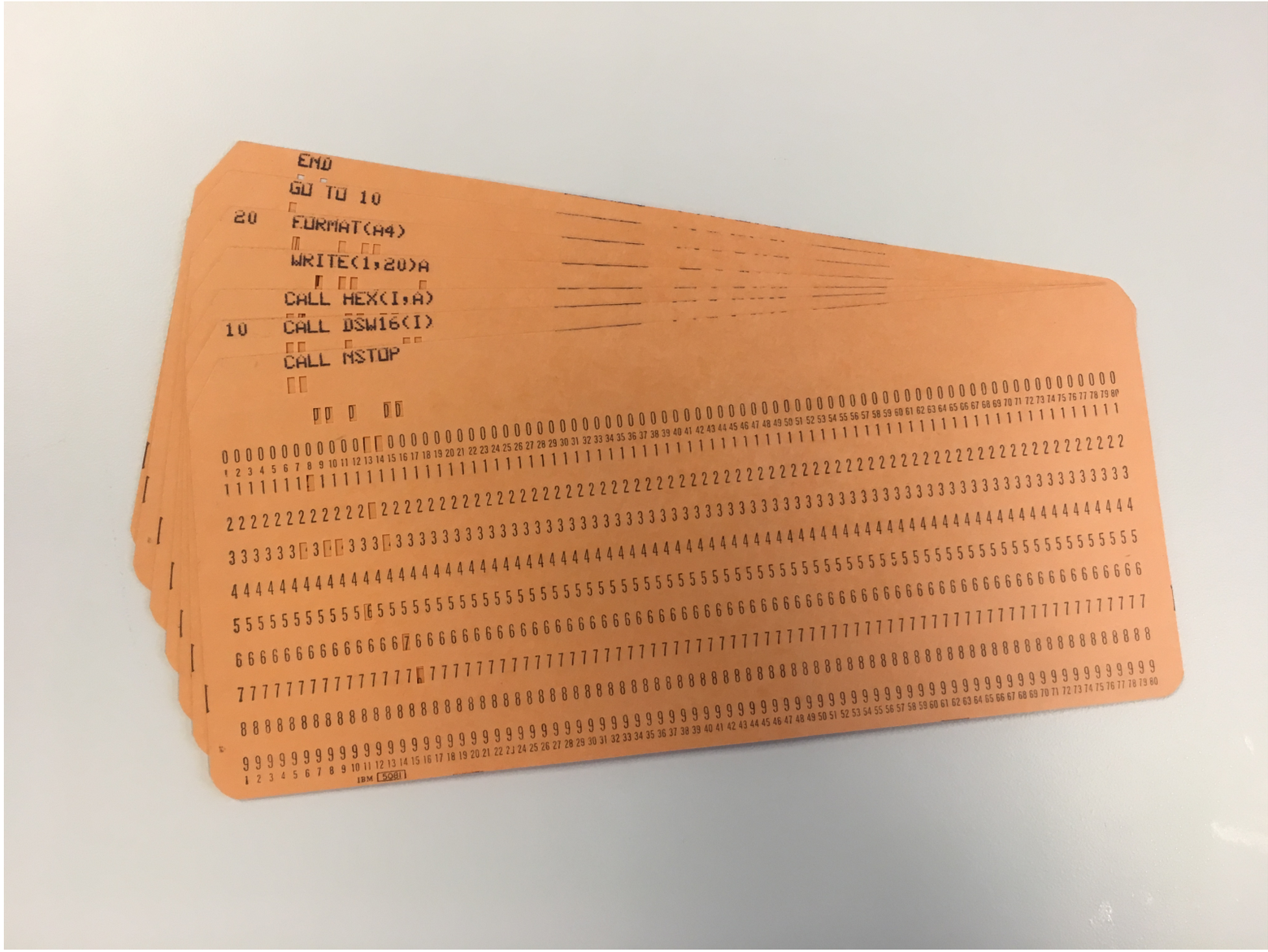
Computers and Hacking: A Brief 50-Year View

Guy L. Steele Jr.
Software Architect, Oracle Labs

HackMIT Keynote
Saturday, September 14, 2019

In November 1968 . . .

When I was 14,
my buddy Al Swide
showed me a
Fortran program
he had written,
much like this one:



Boston Latin School Had an IBM 1130

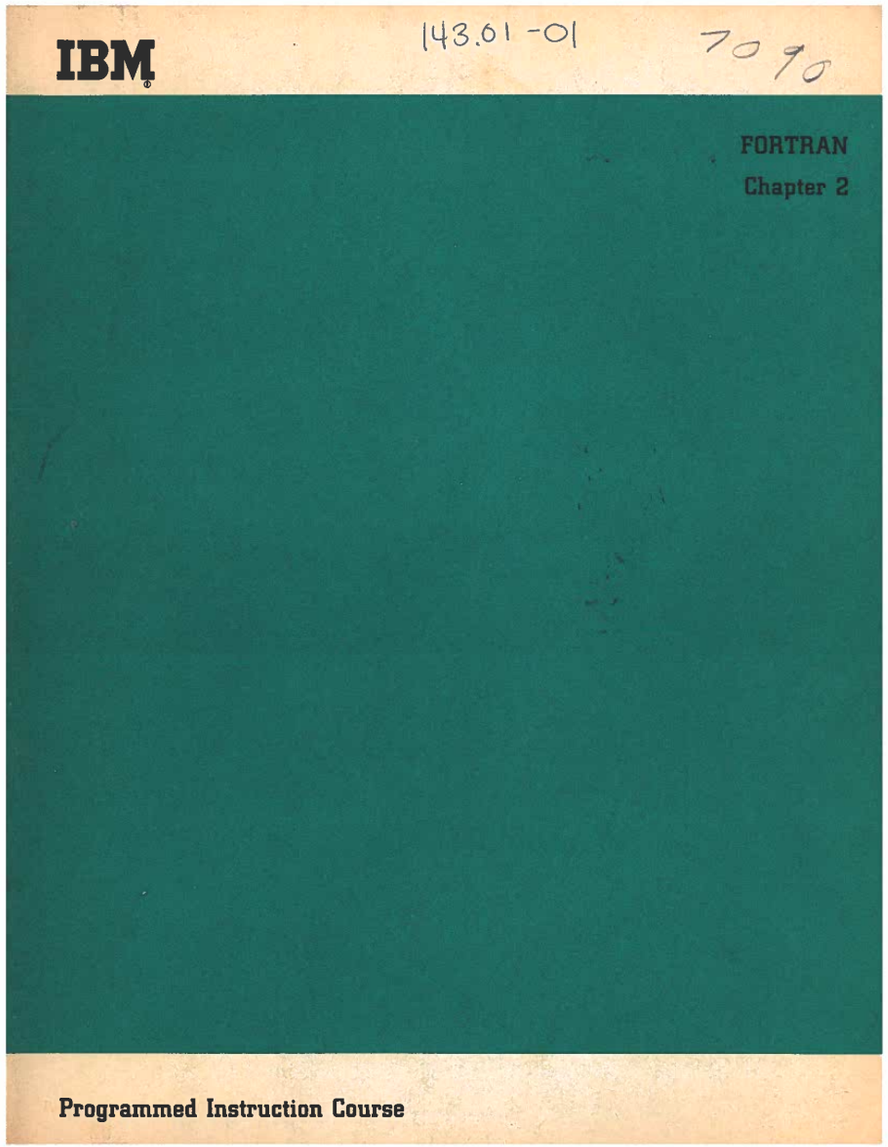
- 8 kilobytes of memory
- 1 megabyte of disk storage
- memory cycle time 3.6 μs
  (“clock speed” $\sim$ 277 kilohertz)
- $41,000 (in 1965)
  (about $330,000 today)

http://ed-thelen.org/comp-hist/vs-ibm-1130.jpg

(Don’t Forget the IBM 1442 Card Read Punch)

Photo by Mike Ross

I Wanted to Learn Fortran



2.6 In FORTRAN programs any statement may be assigned a number. This number is arbitrarily chosen by the program writer and is placed to the left of the actual statement, as, for example:

```
100   Y = A+3.*B
  1   X = X+Y
```

Q. The second statement shown above has a statement number of_____. A. 1

2.7 The rules of statement numbering are simple: any statement may have an assigned number, no particular order of numbers is required, and, naturally, no two statements may have the same number.

Q. Statement numbers are arbitrarily assigned numbers appearing on the_____of the statement to which they refer. A. left

2.8 Given the statements

```
  1   X = 2.1059
  3   A = 3.
3000  B = 4.
  2   Y = A*X**2+B*X
```

the computer will proceed to execute them in the order written: 1, 3, 3000, and then 2.

Q. (True or False) The numerical value of a statement number has no bearing on the order of execution:_____. A. True

2.9 While any statement may have an arbitrarily assigned statement number, it is used in most cases only where a "label" is needed; that is, a statement number is used where it is necessary to refer to that statement from some other part of the program.

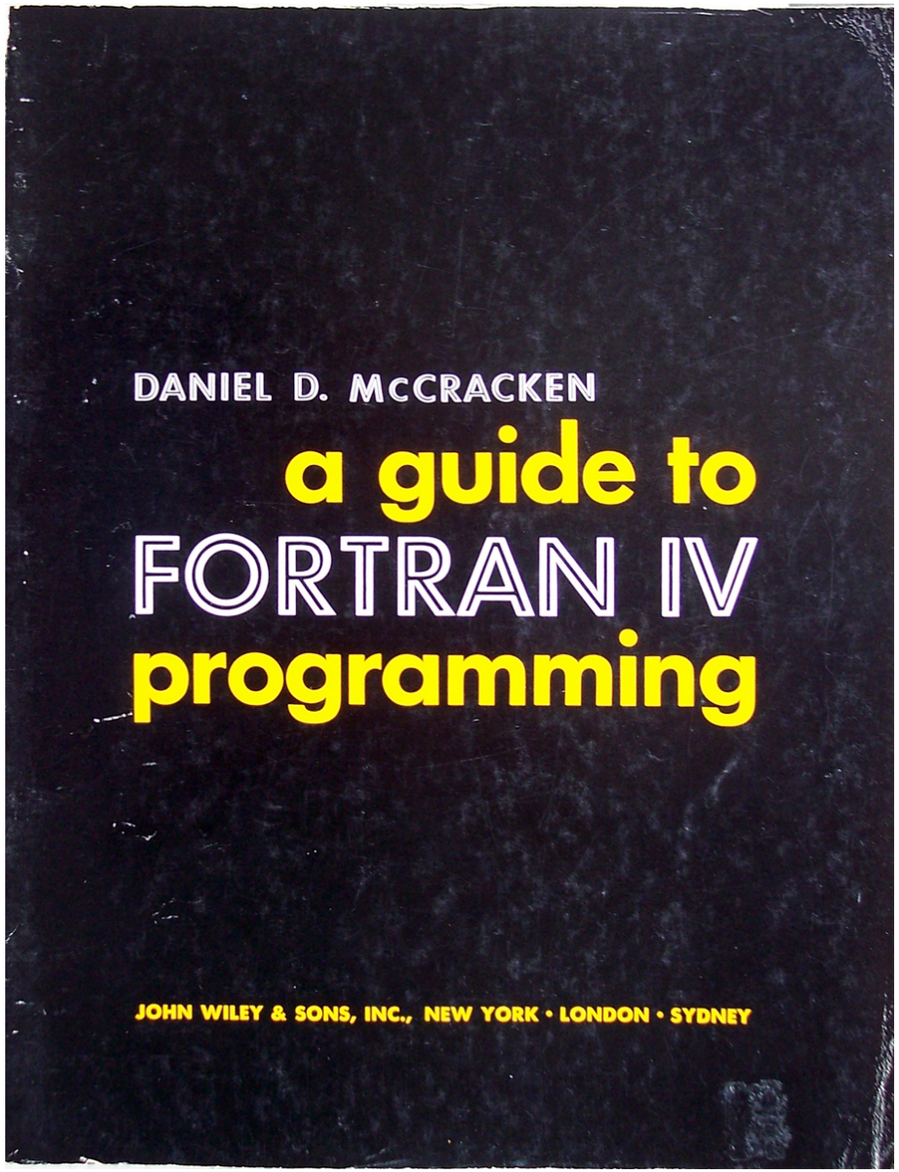
Q. (True or False) Arithmetic Formula statements are the only statements which can have statement numbers:_____. A. False

2.10 Statement numbers are chosen in an arbitrary fashion, but they must not be larger in size than a certain upper limit. On the 7090, for example, statement numbers may not exceed the value of 32767.

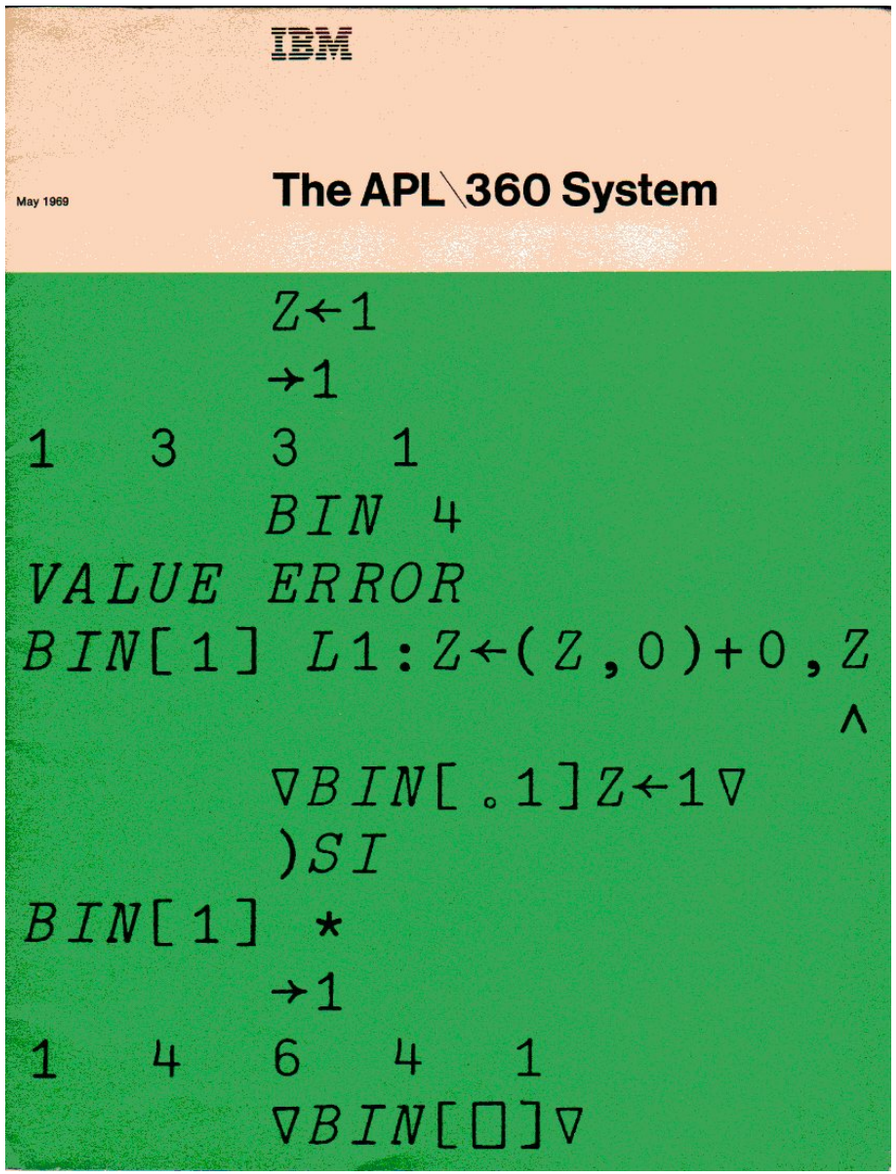
Q. 33766 (is or is not)_____a legal statement number in 7090 FORTRAN. A. is not (33766 is larger than 32767)

2

More Fortran—and Assembly Language



1969 Spring Joint Computer Conference: APL

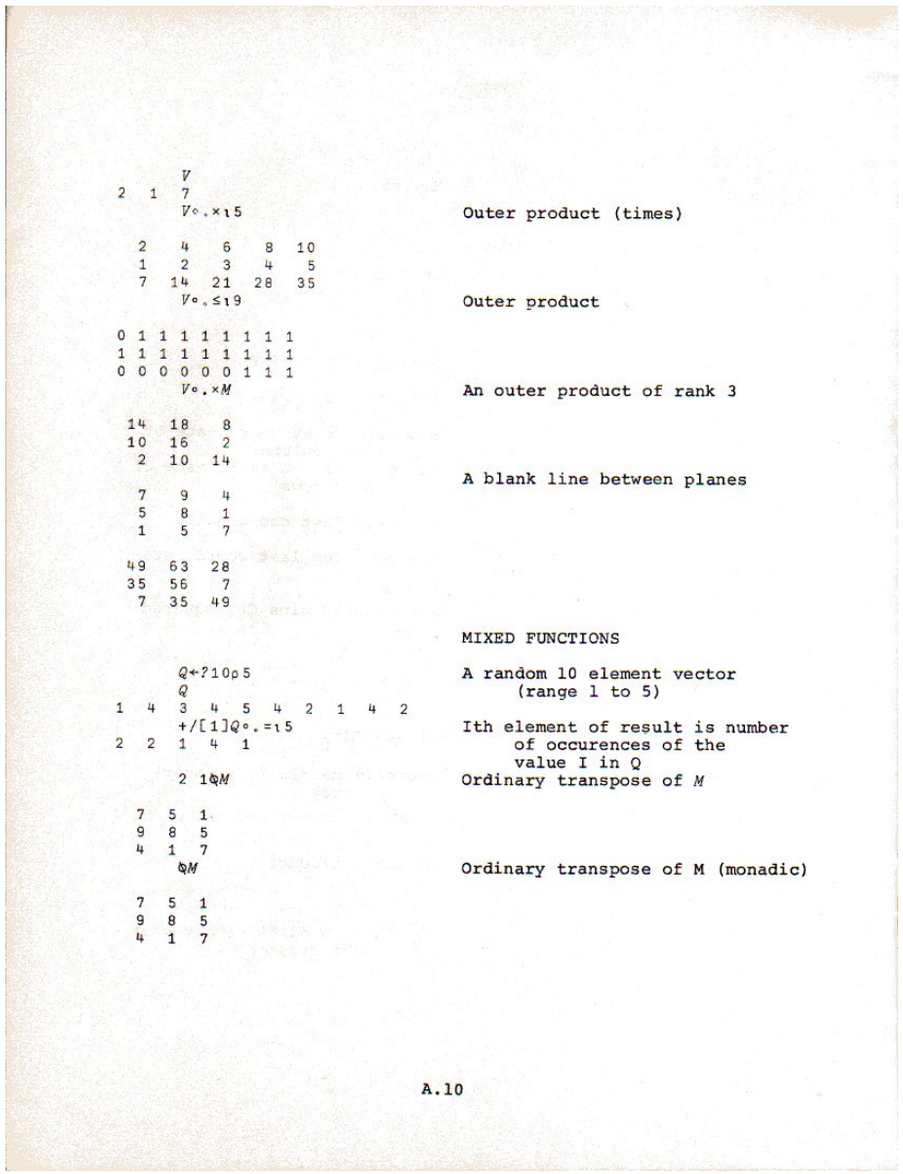


IBM

May 1969

The APL\360 System

```
      Z←1
      →1
1  3  3  1
      BIN 4
VALUE ERROR
BIN[1] L1:Z←(Z,0)+0,Z
                    ∧
      ∇BIN[.1]Z←1∇
      )SI
BIN[1] *
      →1
1  4  6  4  1
      ∇BIN[⎕]∇
```



```
      V
2  1  7
      V∘.×ι5                    Outer product (times)

   2   4   6   8  10
   1   2   3   4   5
   7  14  21  28  35
      V∘.≤ι9                    Outer product

0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1
      V∘.×M                     An outer product of rank 3

 14  18   8
 10  16   2
  2  10  14

  7   9   4                     A blank line between planes
  5   8   1
  1   5   7

 49  63  28
 35  56   7
  7  35  49


                                MIXED FUNCTIONS

      Q←?10ρ5                   A random 10 element vector
      Q                              (range 1 to 5)
1  4  3  4  5  4  2  1  4  2
      +/[1]Q∘.=ι5               Ith element of result is number
2  2  1  4  1                        of occurences of the
                                      value I in Q
      2 1⍉M                     Ordinary transpose of M

  7  5  1
  9  8  5
  4  1  7
      ⍉M                        Ordinary transpose of M (monadic)

  7  5  1
  9  8  5
  4  1  7
```

A.10

Hack: Making Music

technikum29 computer museum https://www.technikum29.de/shared/photos/rechnertechnik/ibm-1130-konsole.jpg

Hack: Making Music

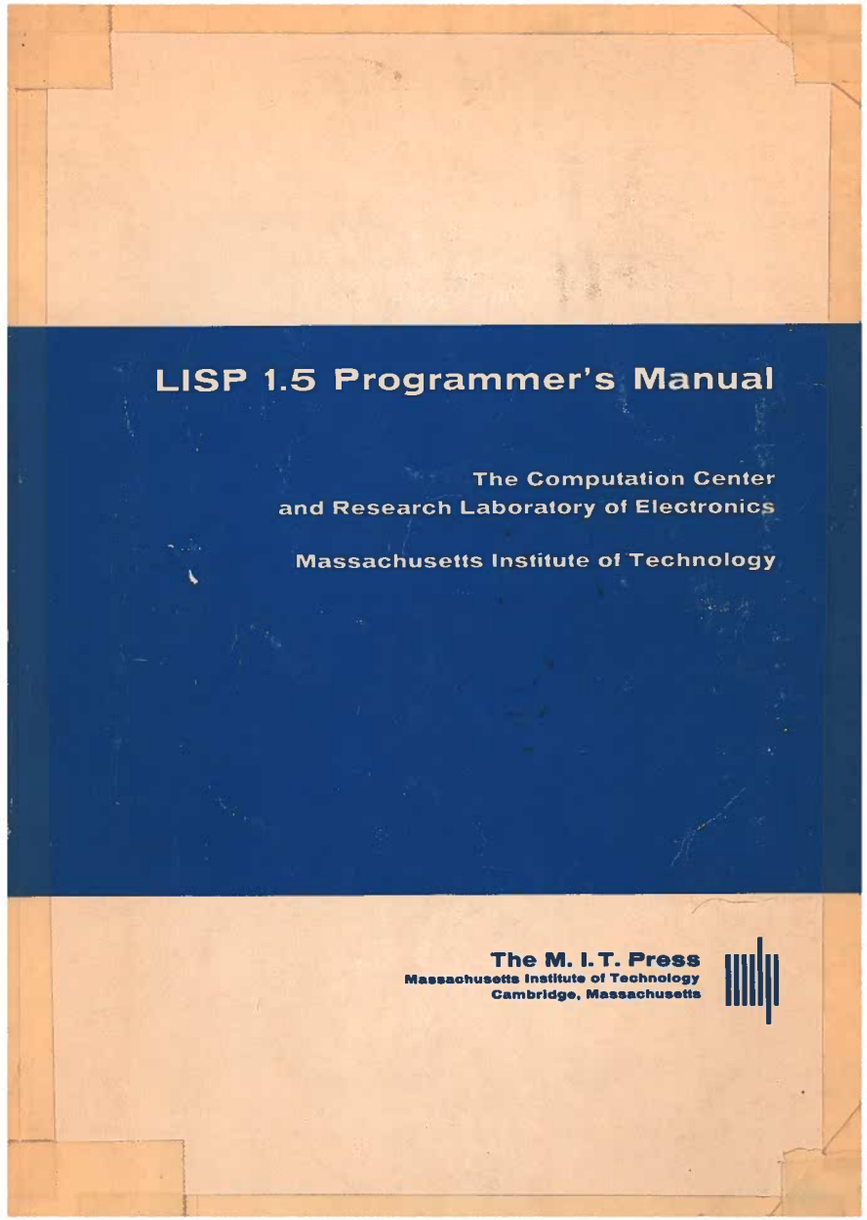
technikum29 computer museum https://www.technikum29.de/shared/photos/rechnertechnik/ibm-1130-konsole.jpg

Hanging Out at MIT (1969–1972)

- High School Studies Program (MIT students teaching high school students)
  - Group theory
  - Programming courses
  - Access to another IBM 1130!
- MIT AI Lab and Project MAC
  - Technical reports free for the taking
  - Access to a PDP-10 computer! (1 megahertz, 1 megabyte, 1 megabuck)
  - Access to LISP documentation and source code
- Digital Equipment Corporation field office (Central Square)
  - Hardware and software manuals free for the taking
- MIT Press book sales

Lisp Took Me a While to Figure Out

The *LISP 1.5 Programmer's Manual* gave a definition of the Lisp programming language in terms of itself. This confused me, and I was convinced that this sort of recursive definition must be totally broken.* I had a chip on my shoulder about Lisp for the next couple of years, which I had to work to overcome.



* Turns out it *was* indeed *slightly* broken, as John Reynolds explained in 1972.

To Make a Long Story Short . . .

- 1971–1972: I implemented the Lisp language on the IBM 1130
  - Design and documentation loosely based on MIT's MacLISP
  - But I added a character-string data type
- May 1972: Graduated from Boston Latin

  Parents immediately said:

To Make a Long Story Short . . .

- 1971–1972: I implemented the Lisp language on the IBM 1130
  - Design and documentation loosely based on MIT's MacLISP
  - But I added a character-string data type
- May 1972: Graduated from Boston Latin

  Parents immediately said: **Get a job!**

To Make a Long Story Short . . .

- 1971–1972: I implemented the Lisp language on the IBM 1130
  - Design and documentation loosely based on MIT's MacLISP
  - But I added a character-string data type
- May 1972: Graduated from Boston Latin
  Parents immediately said: **Get a job!**
- July 1972: Found a job—at MIT! Hacking Lisp!

To Make a Long Story Short . . .

- 1971–1972: I implemented the Lisp language on the IBM 1130
  - Design and documentation loosely based on MIT's MacLISP
  - But I added a character-string data type
- May 1972: Graduated from Boston Latin
  Parents immediately said: **Get a job!**
- July 1972: Found a job—at MIT! Hacking Lisp!
- 1972–1975: Undergraduate at Harvard while working at MIT
- 1975–1980: Graduate student at MIT in computer science

Some Projects and Languages I Have Worked On

- Scheme
- EMACS
- Common Lisp
- C compilers
- data parallel programming
- High Performance Fortran
- Java
- Fortress
- *The Hacker's Dictionary*
  aka "the Jargon File"

Books I have co-authored:

Observation 1

In 1969, *all* computers were expensive.

Today, the big ones are still expensive, but reasonable ones are way cheap.

Observation 2

In 1969, computers were *institutional* devices.

Today, most computers are *personal* devices.

(Nevertheless, much of their usefulness comes from interaction with institutional computers!)

Observation 3

In 1969, access to computers was difficult.
I *dreamed* of having a computer in my basement.

Two decades later, I bought my own computer
—and a laser printer!
(I could have had a small car for the same price.)

Today—well, you know.

Laptops (high hundreds of dollars)

https://upload.wikimedia.org/wikipedia/commons/9/9a/MacBook_Pro.jpg

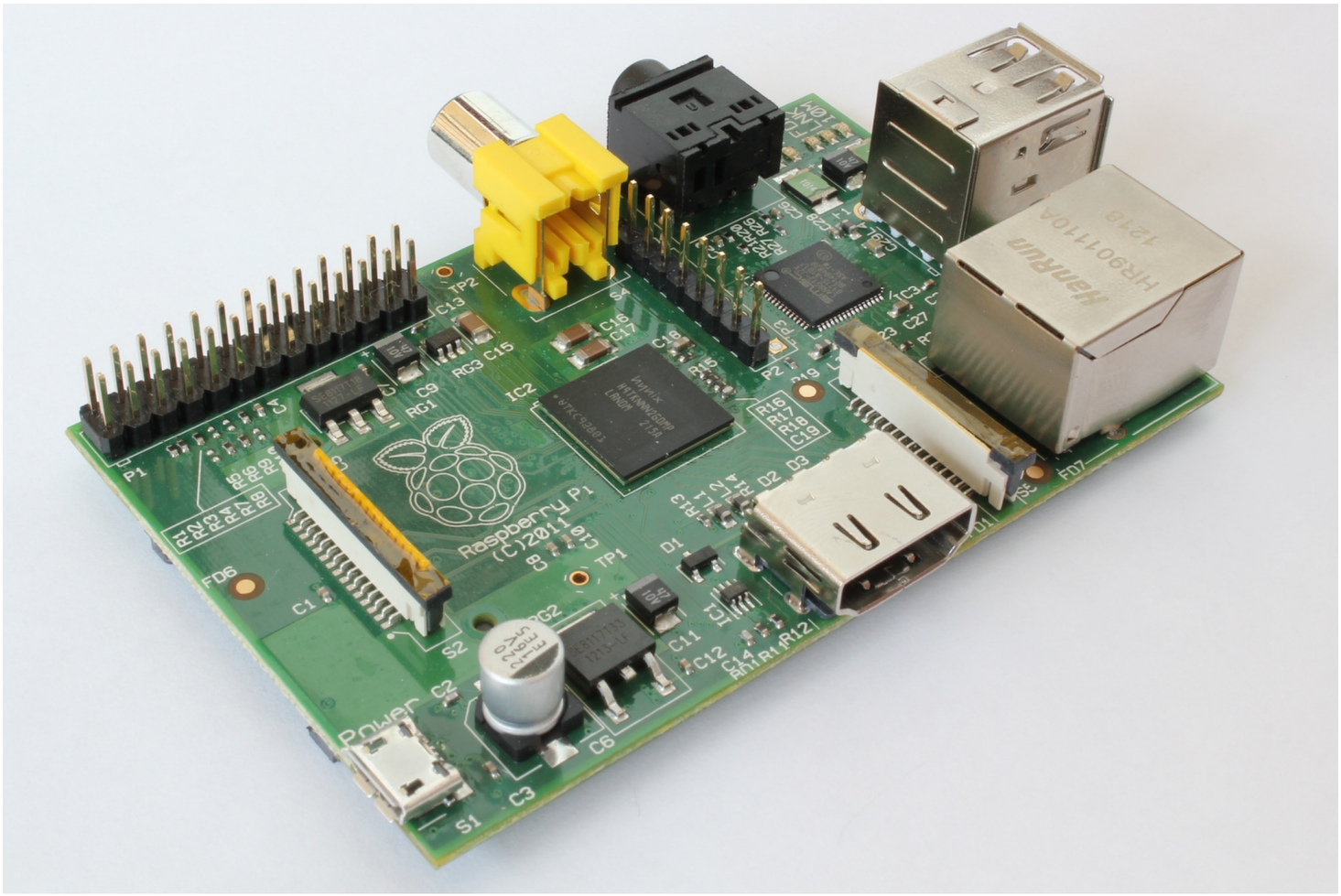
https://upload.wikimedia.org/wikipedia/commons/b/b4/Dell_Inspiron_1525_250618.jpg

Phones (low hundreds of dollars)

https://upload.wikimedia.org/wikipedia/commons/f/fd/Apple_iPhone.jpg

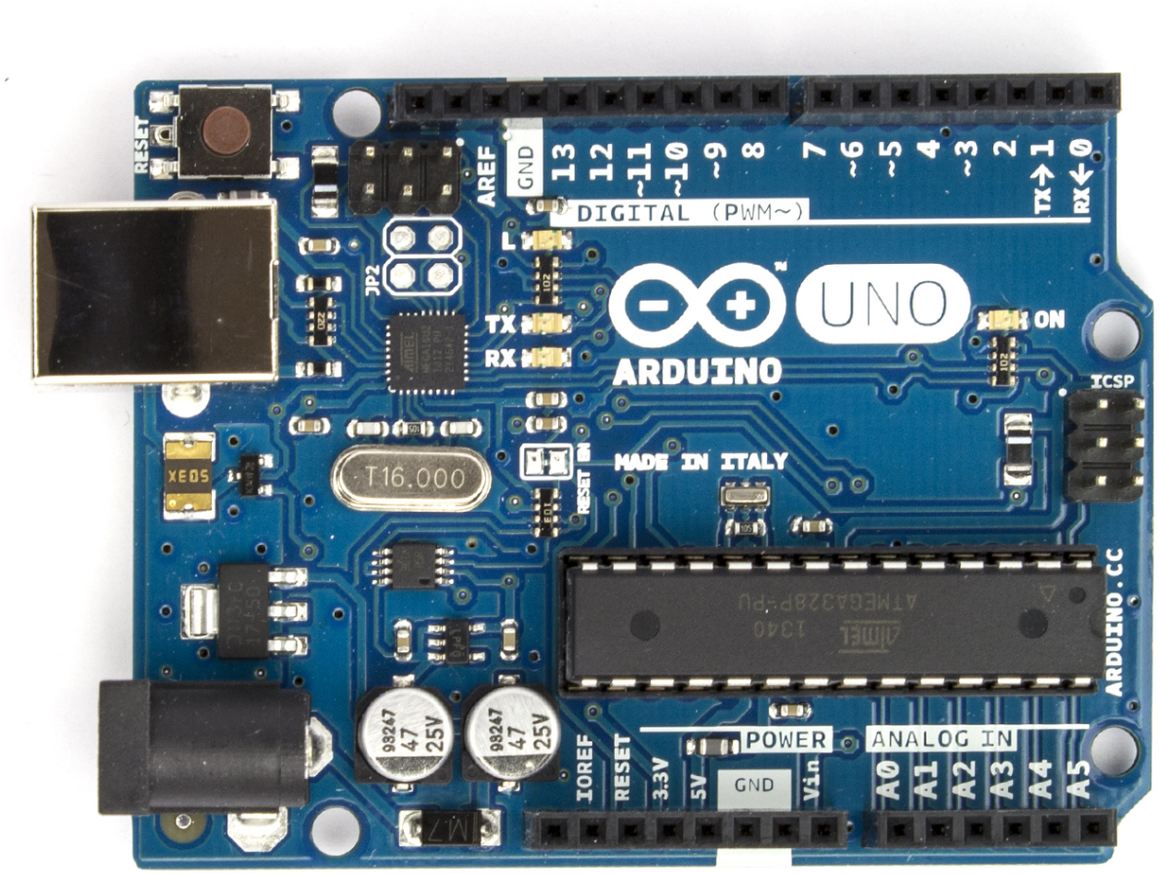
https://upload.wikimedia.org/wikipedia/commons/1/16/Android_Smartphones.jpg

Raspberry Pi (well under 50 bucks; gigahertz, gigabytes)



https://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

Arduino (well under 25 bucks; megahertz, kilobytes)



https://upload.wikimedia.org/wikipedia/commons/a/a6/Arduino_Uno_006.jpg

Observation 4

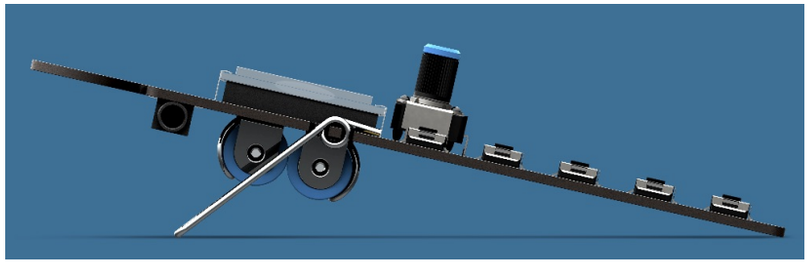
Moore's Law lasted for most of my career.
(Transistors on a chip doubling every 2 years, and CPU speed similarly until recently.)

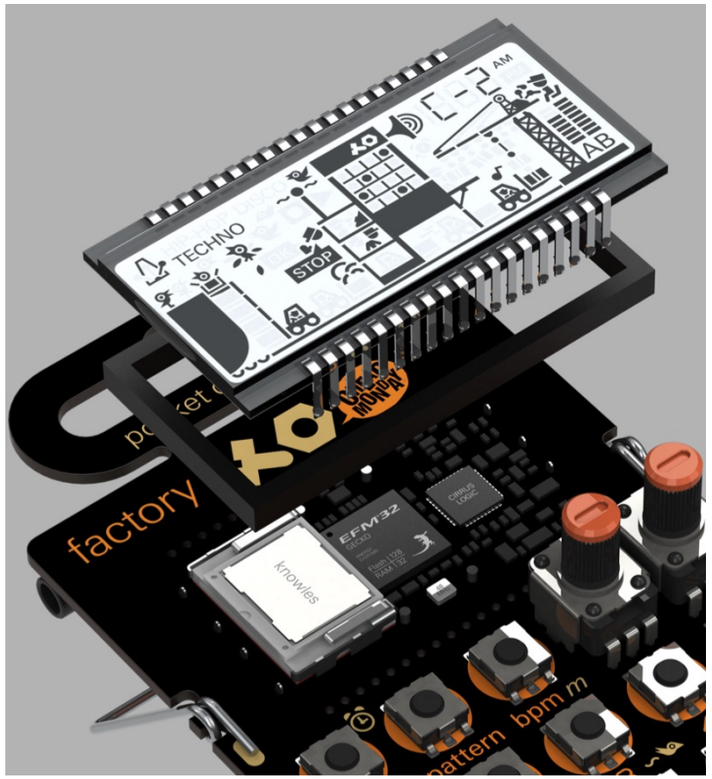
Sometimes I exploited that.

Now highest performance *requires* parallelism.

But many apps don't need highest performance.
Even small computers are fantastically good.

Pocket Operators (line of handheld musical grooveboxes, 50 to 90 dollars)





RAM: 32 kilobytes

Flash: 128 kilobytes

Clock: 48 megahertz

https://teenage.engineering/products/po

Observation 5

In 1969, access to *information* was difficult.
I spent a lot of time and effort to acquire it.

Today, we have the Internet at our fingertips.

The problem is figuring out what to *ignore*.
That, too, takes time and effort.

Words of Wisdom

Make good use of your time.

Don't be *too* distracted by fluff.

Random curiosity is a good thing —but give it guidance and focus.

The best work helps many people.

Enjoy this weekend!

Use your time well.

May you have a fruitful intellectual journey.


ORACLE®