The Legendary IBM 1401

A Major Milestone in the History of Modern Computing

Department of Applied Data Science

Department of Computer Science

Engineering Extended Studies

San José State University

Ron Mak

www.cs.sjsu.edu/~mak

What was the IBM 1401?

A “small scale” computer system developed by IBM in the late 1950s.

What was the IBM 1401?

- How did this small-scale system help free thousands of businesses and institutions from storing and processing data on punched cards?

- What were the unique aspects of its architecture?

- Why are the 1401 system's peripherals (I/O devices) still considered electromechanical marvels today?

- What was it like to program the 1401?
  - We'll do some simple Autocoder programming on a PC-based simulator.

What was Computing Like Before the 1401?

- Business data processing involved applications that manipulated data records:
  - Inventory
  - Billing and receivables
  - Payroll

What was Computing Like Before the 1401?

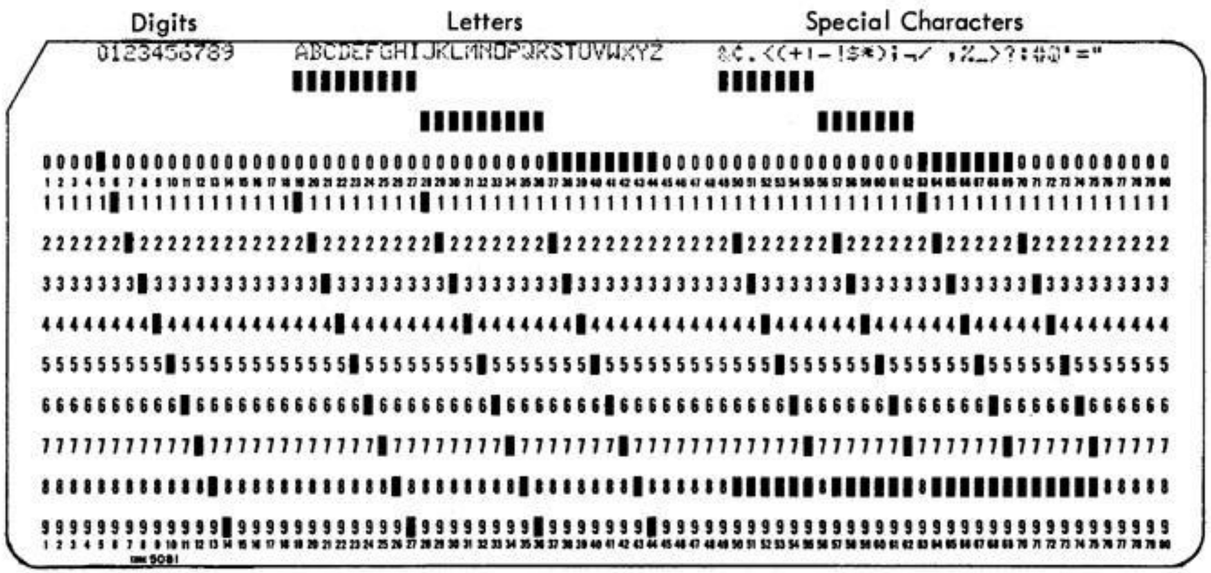
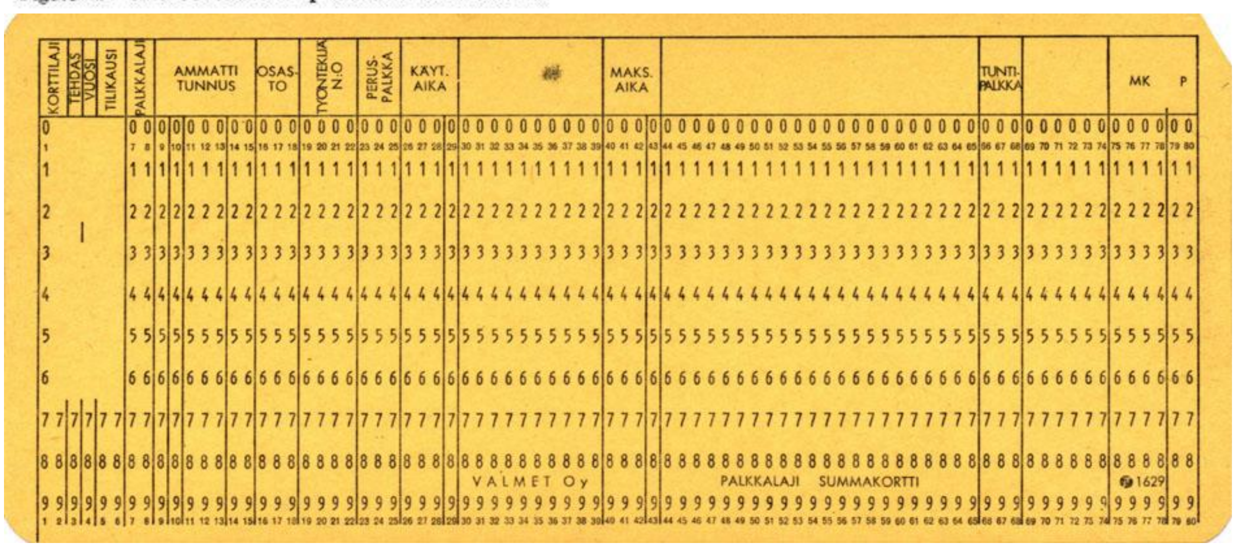


Figure 4. Card Codes and Graphics for 64-Character Set



- Data was stored in punched cards called IBM cards or Hollerith cards
  - Named after Herman Hollerith.
- 80 columns per card, one character per column.
- Up to 12 punched holes per column.
- Alphanumeric data, often grouped into fields.

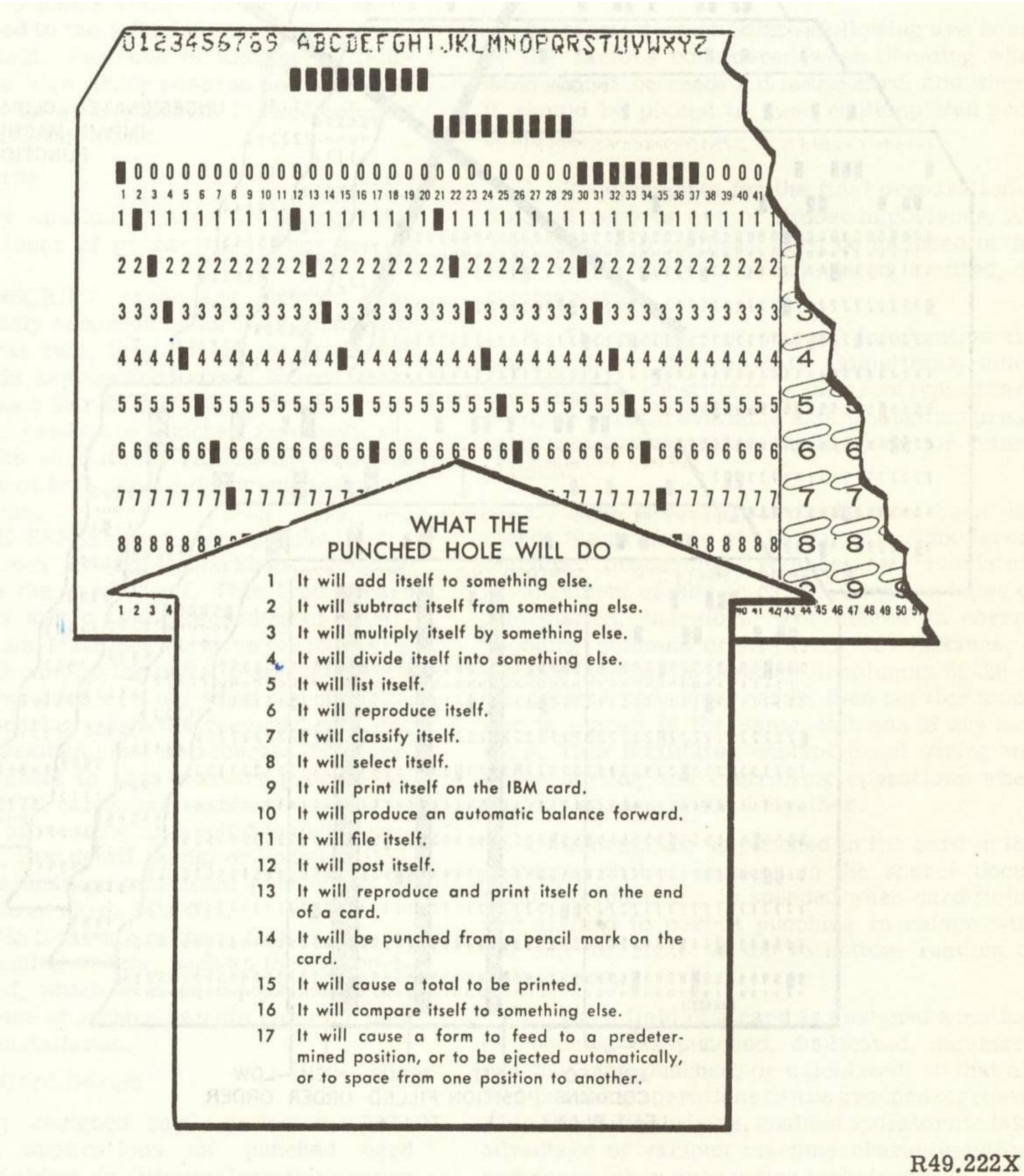


Figure 2-10.—Capabilities of the punched hole.

Punched Cards

- Punched cards used the Hollerith code.
  - Rows 0-9 were numeric punches
  - The topmost row was row 12 and the second row was 11.
  - Rows 12, 11, and 0 were zone punches.
- Examples:

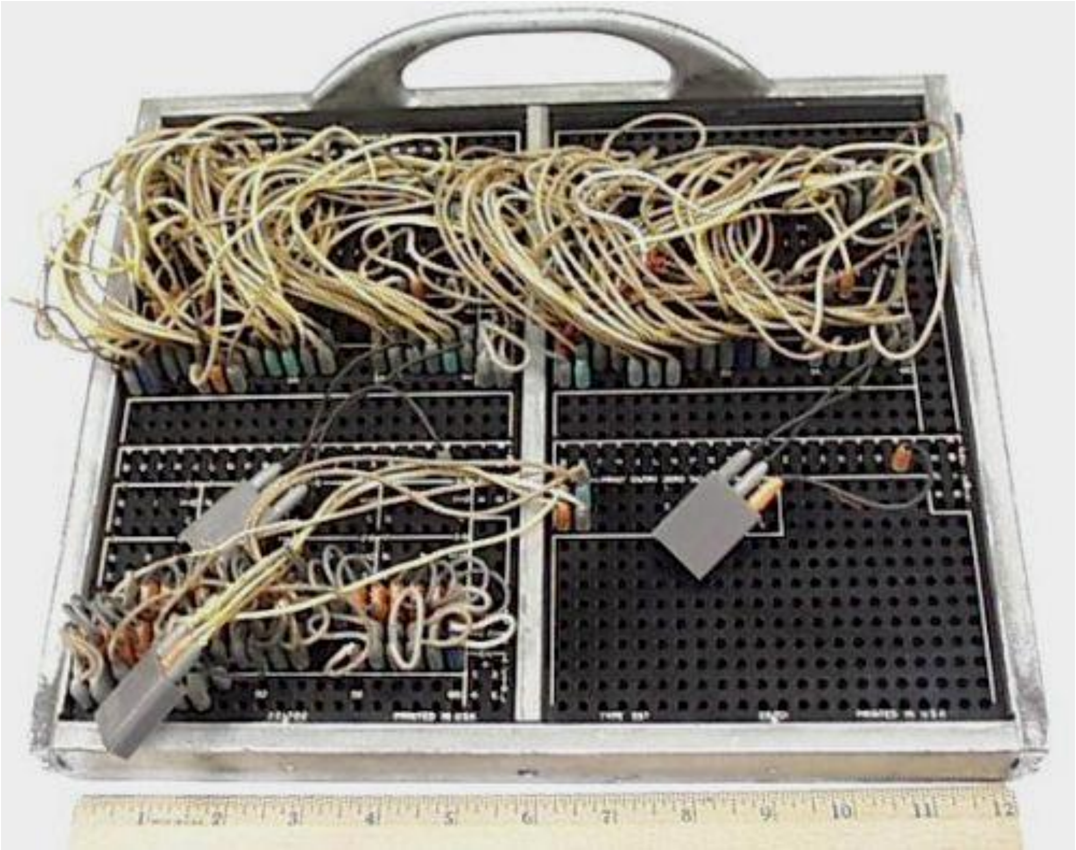
Char	Punch
3	3
A	12-1
M	11-4
S	0-2
$	11-3-8

What was Computing Like Before the 1401?

- A data processing application involved passing decks of punched cards through electromechanical unit-record machines.

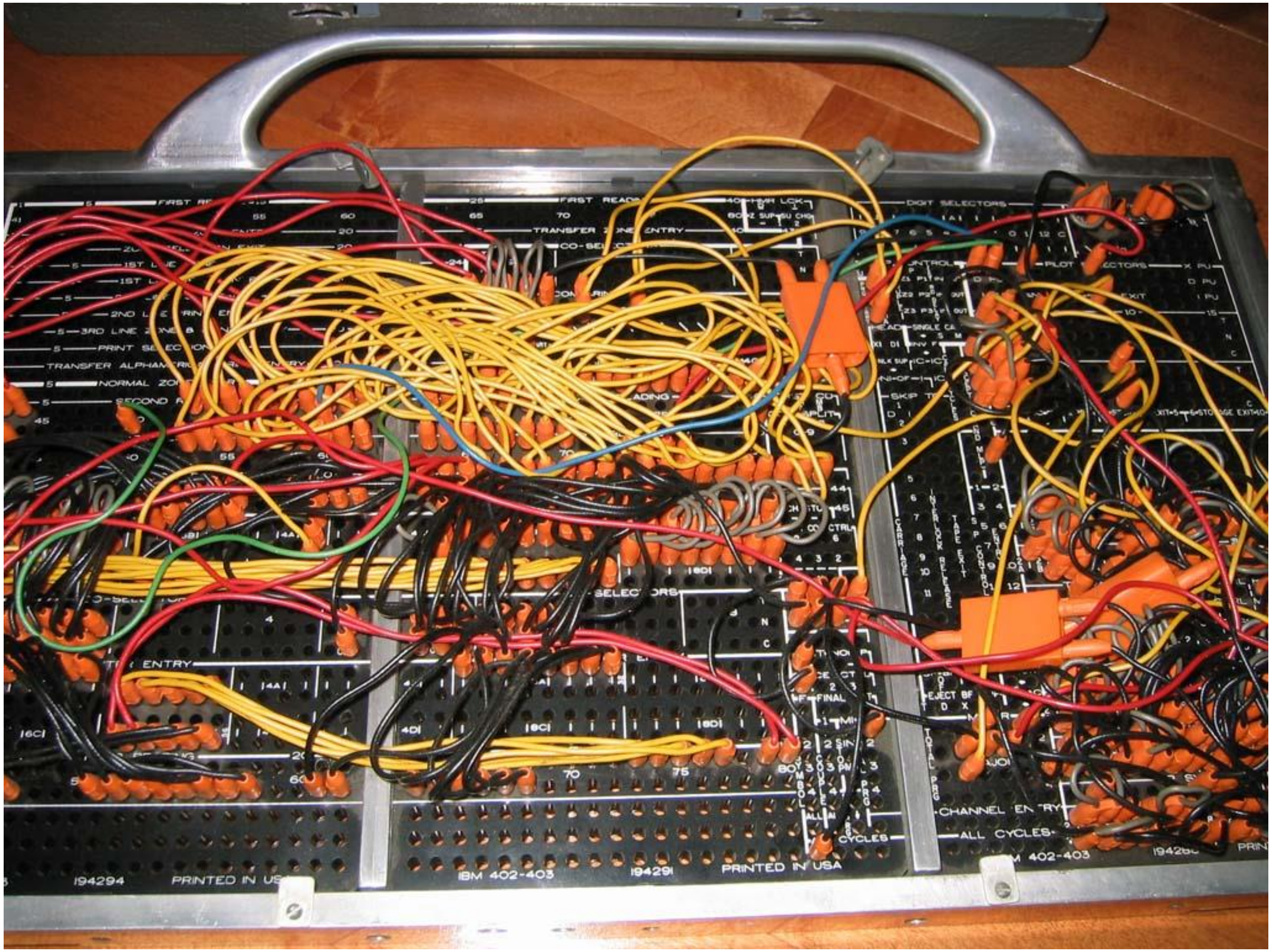
- Repetitive sort, calculate, collate, and tabulate operations ...
  - ... were programmed with hand-wired plugboard control panels.

Plugboard Control Panel



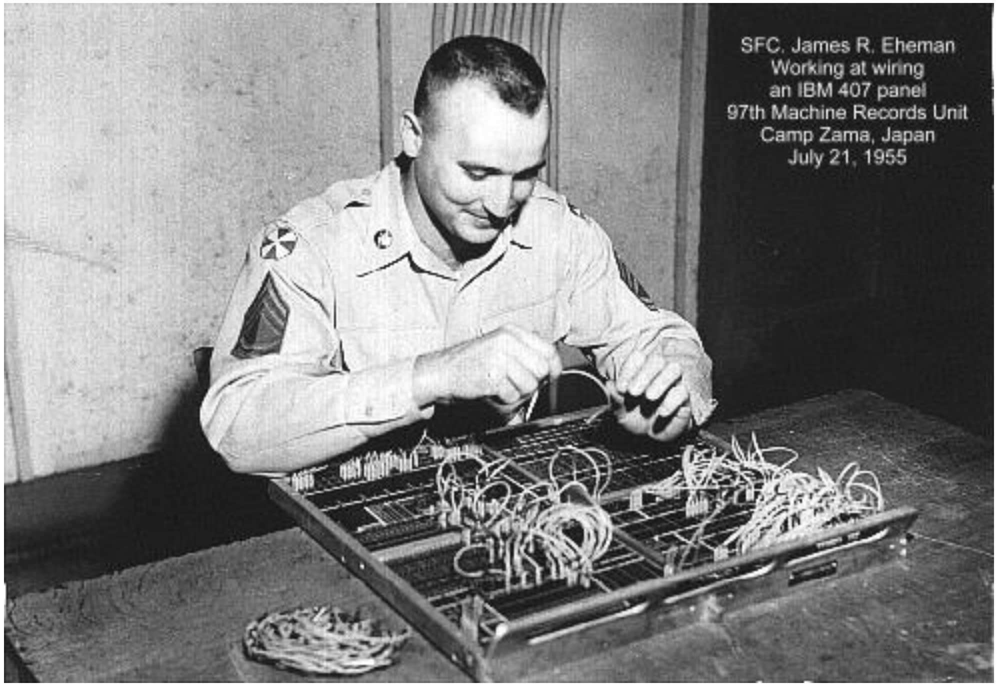
IBM 407 Accounting Machine (1949)

Plugboard Control Panel



Programming a Plugboard

- “Programming” was hand-wiring plugboards.



“Hmm, should I pass this parameter by value or by reference?”

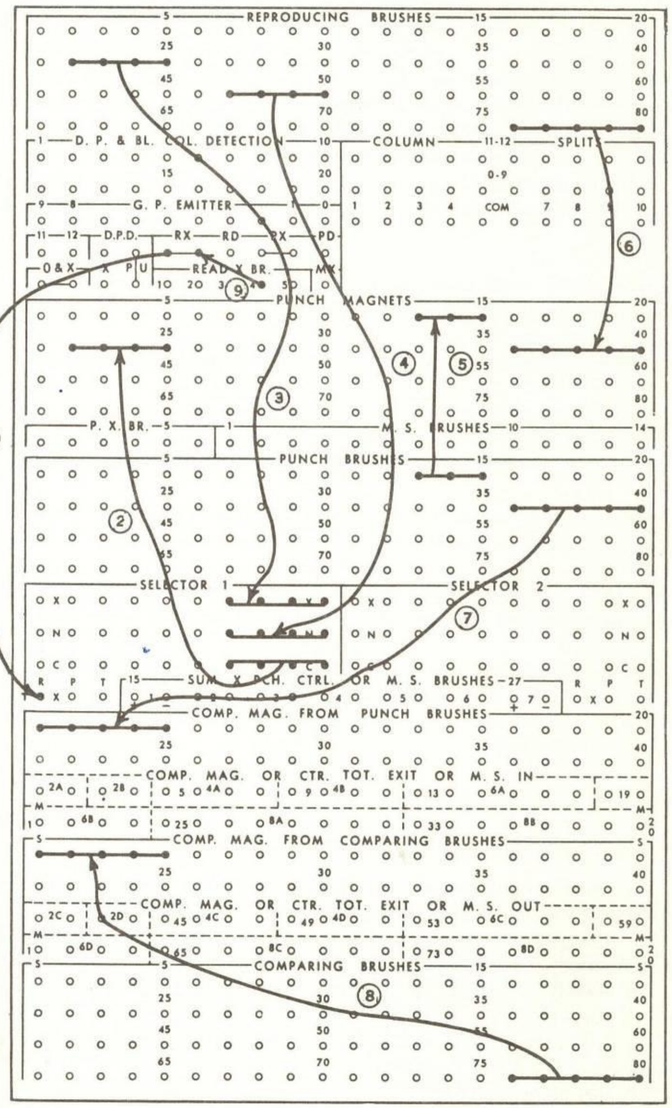


Figure 3.33 Plugboard wiring for IBM 514

Programming a Plugboard

- Plugboard wiring diagram
  - It doesn’t look too complicated, does it?

Data Processing

- Data processing was all about <u>punched cards</u>.

- My school compiler project:
  - 3½ boxes of punched cards
  - Each box = 2000 cards, 10 lbs.

Data Processing

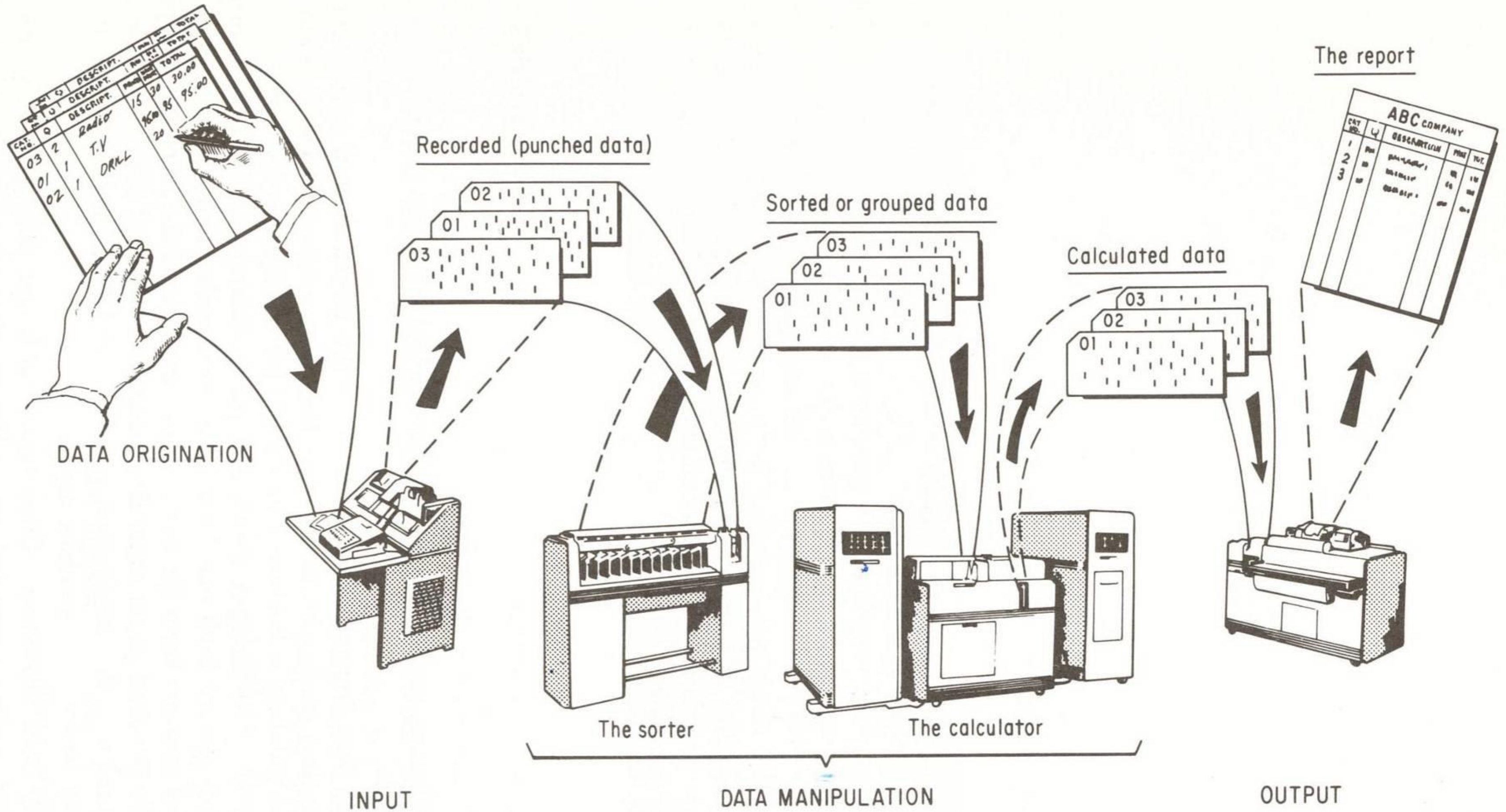


FIG. 6-1 The punched-card data processing cycle

Data Processing

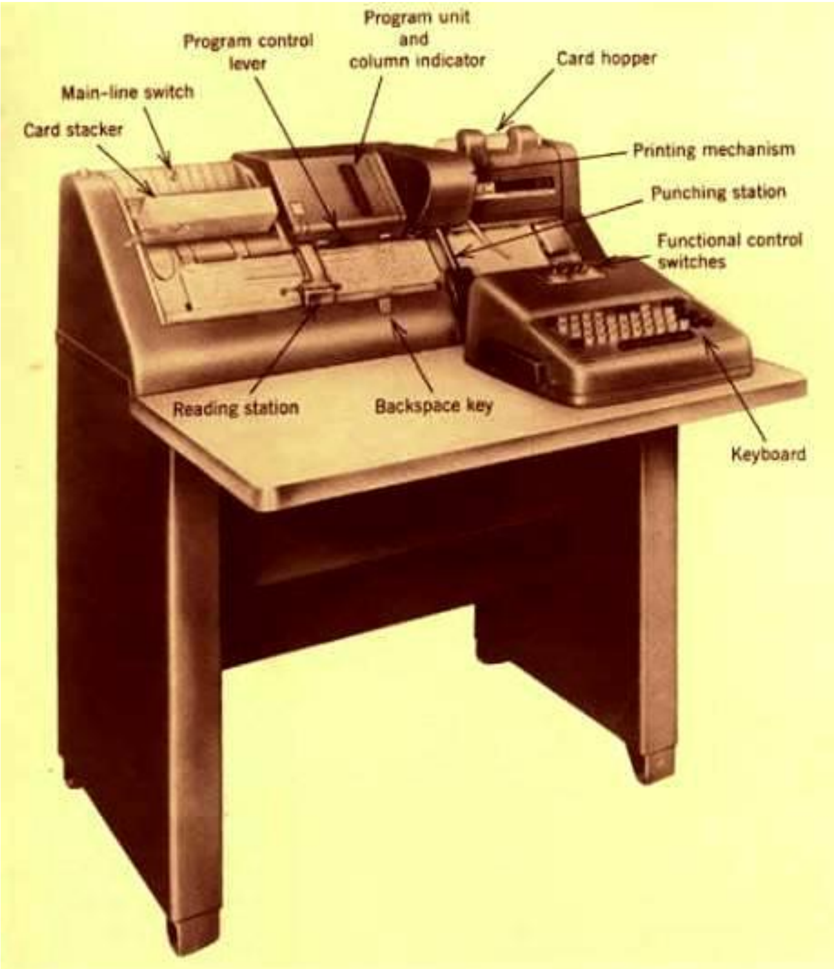


FIGURE 4.11 IBM 026 Keypunch (Courtesy IBM).

- Cards were punched manually at a keypunch machine.
  - Or they were punched automatically by unit-record equipment under program control.

Data Processing

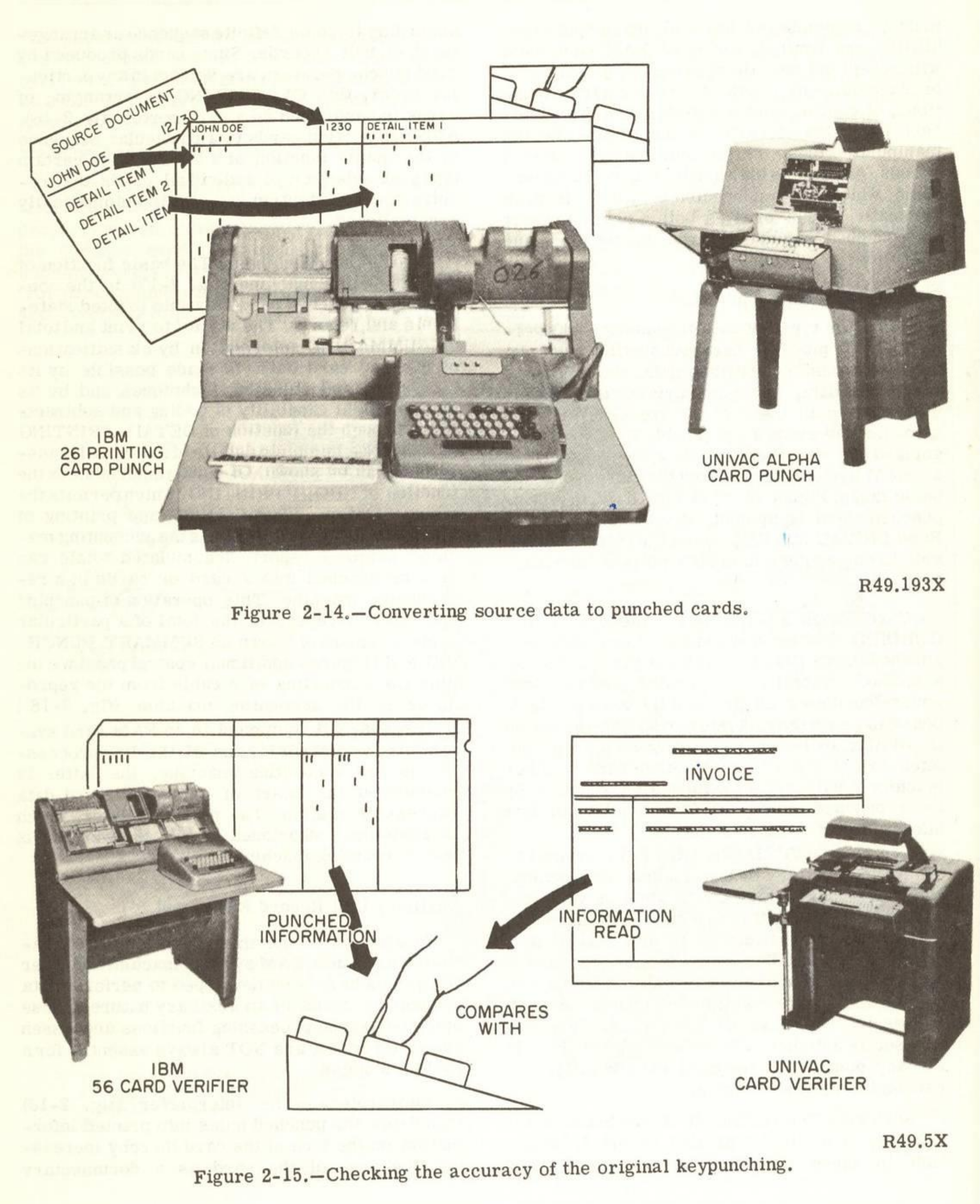


Figure 2-14.—Converting source data to punched cards.

Figure 2-15.—Checking the accuracy of the original keypunching.

- Cards were re-keyed on a verifier to ensure accuracy.
  - Good cards were notched at the top right edge.
  - Bad cards were notched at the top edge above each erroneous column.

Data Processing

- A sorter sorted cards one column at a time.
  - You had to run decks of cards multiple times through a sorter.
- Accounting machines performed arithmetic on card fields and printed reports.

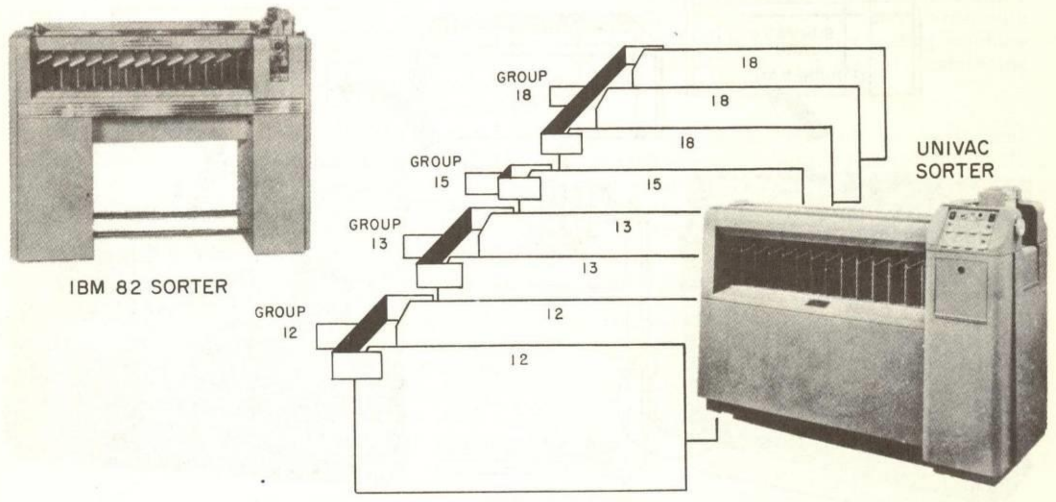


Figure 2-16.—Grouped cards in a definite sequence.

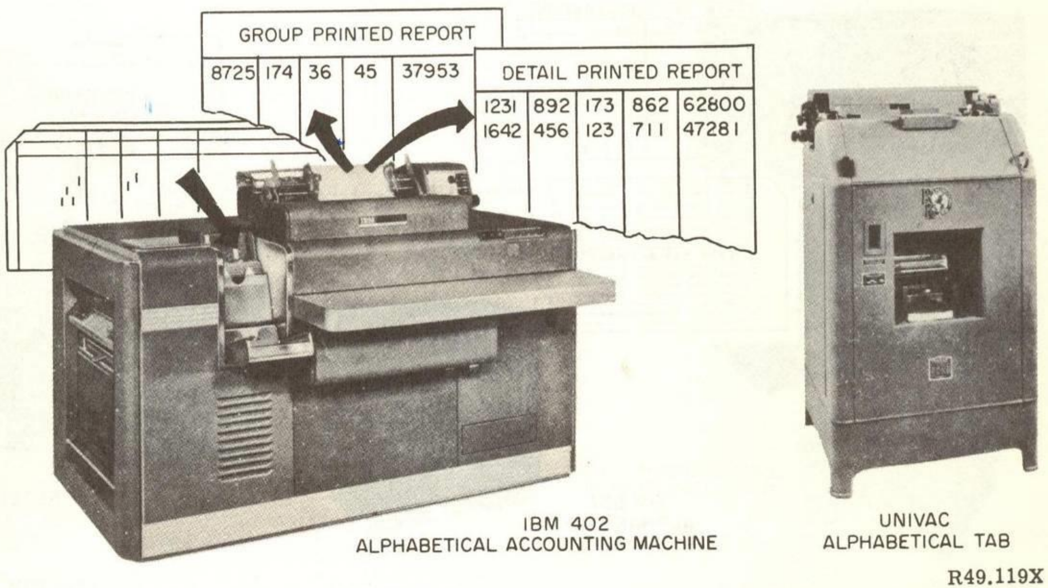


Figure 2-17.—End of the line processing.

Data Processing

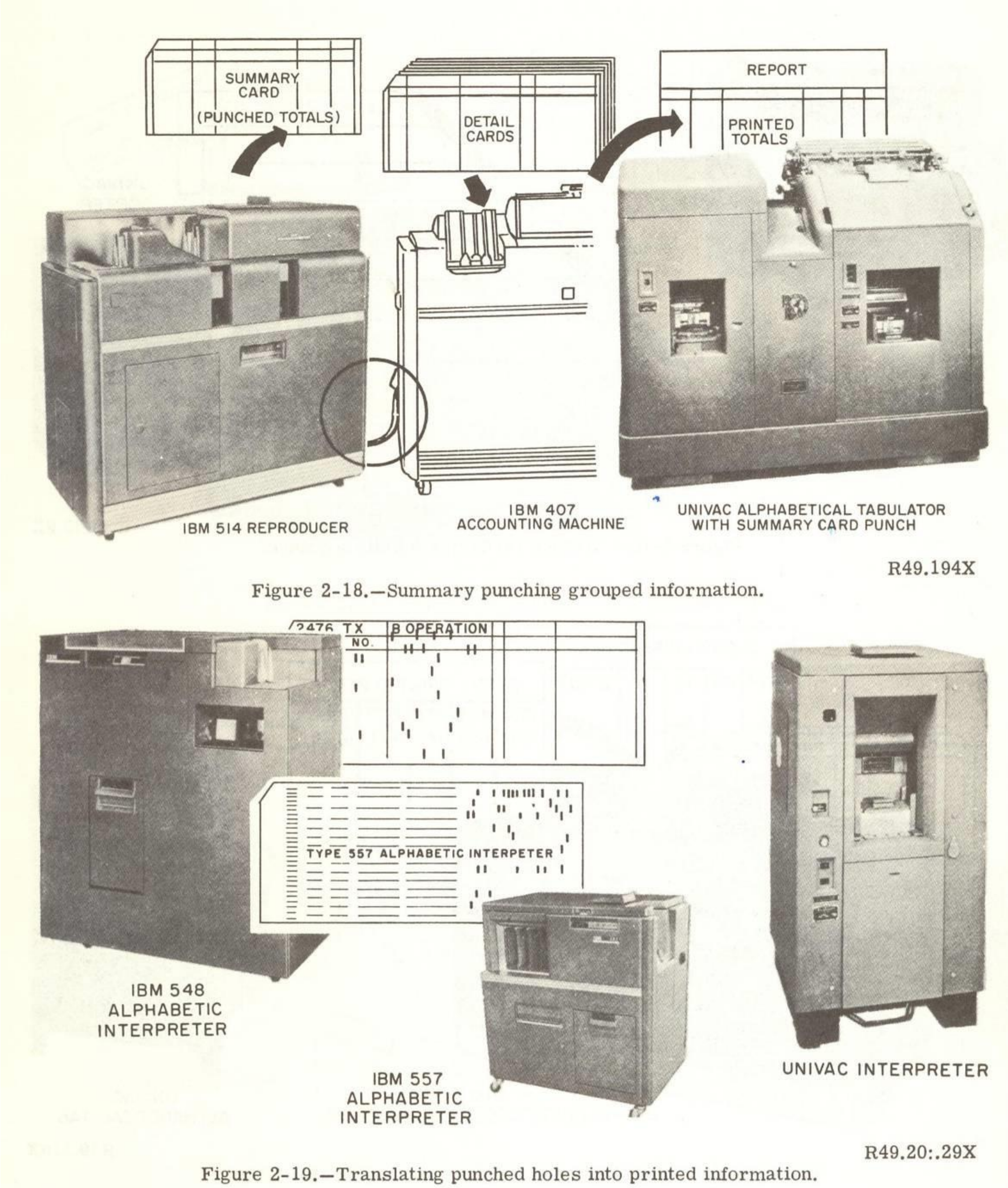


Figure 2-18.—Summary punching grouped information.

Figure 2-19.—Translating punched holes into printed information.

- Reproducers made copies of card decks.
- Tabulators were accounting machines: simple arithmetic plus printing.
- Interpreters read cards and printed information on the cards.

Data Processing

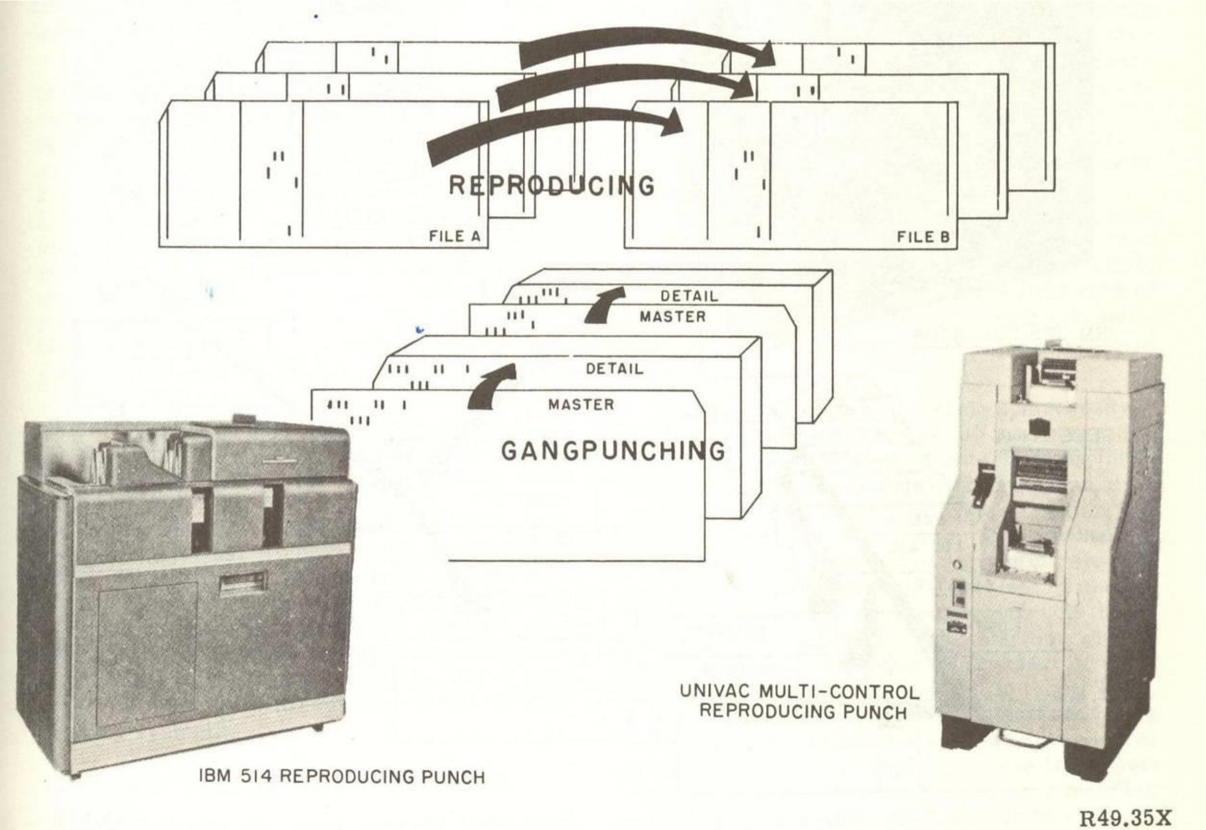


Figure 2-20.—Automatic reproducing and gang punching.

- Gang punching: Automatically punch multiple cards from the contents of a single card.

Data Processing

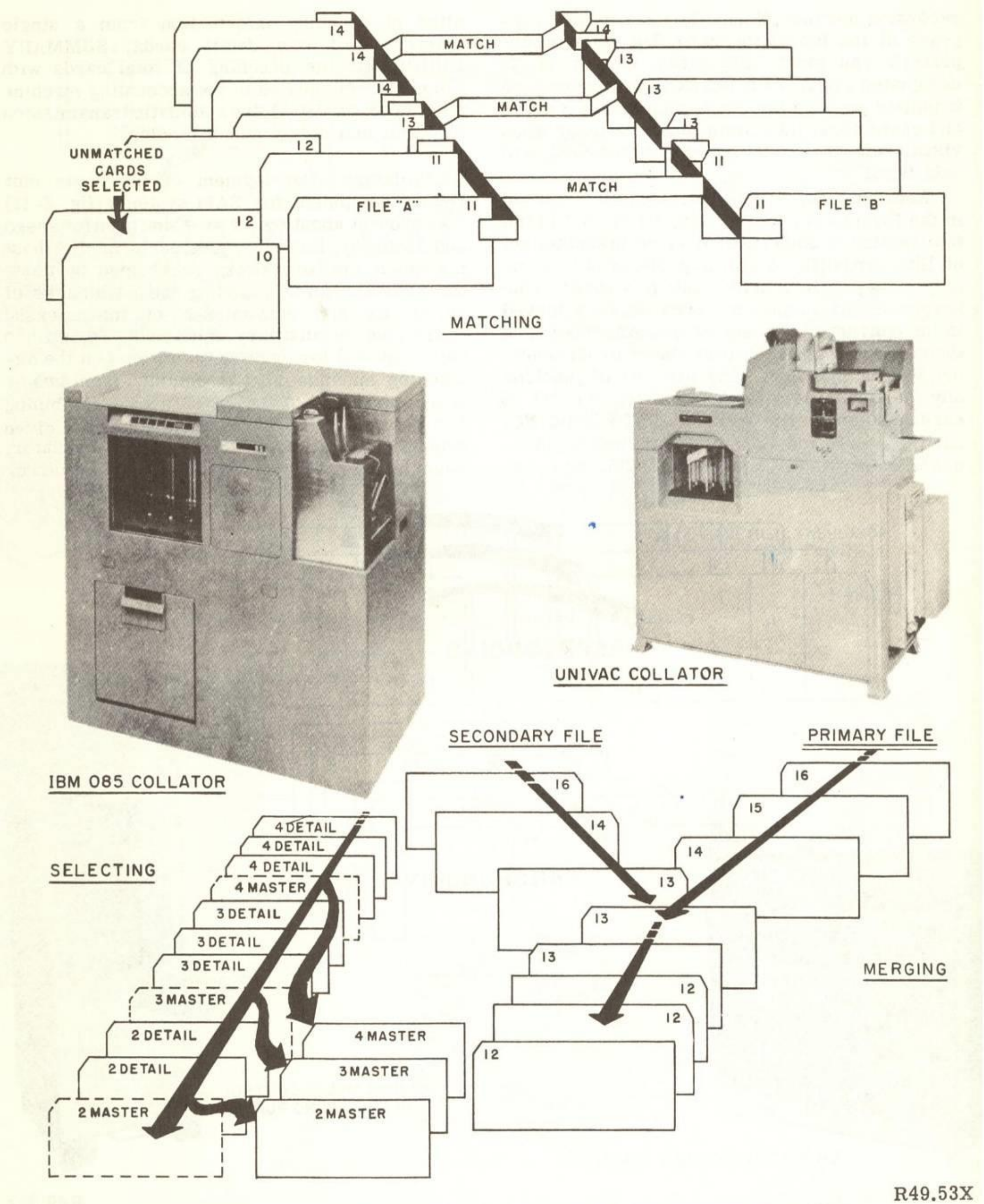


Figure 2-21.—Filing machines that arrange cards for subsequent operations.

- A collator compared and merged decks of punched cards.

Running a Data Processing Application ...

- ... meant passing decks of cards through a sequence of unit-record machines.
  - Each machine was programmed via its plugboard to perform its task for the application.
  - Each machine had little or no memory.
  - The punched cards stored the data records
  - The data records moved as the cards moved.

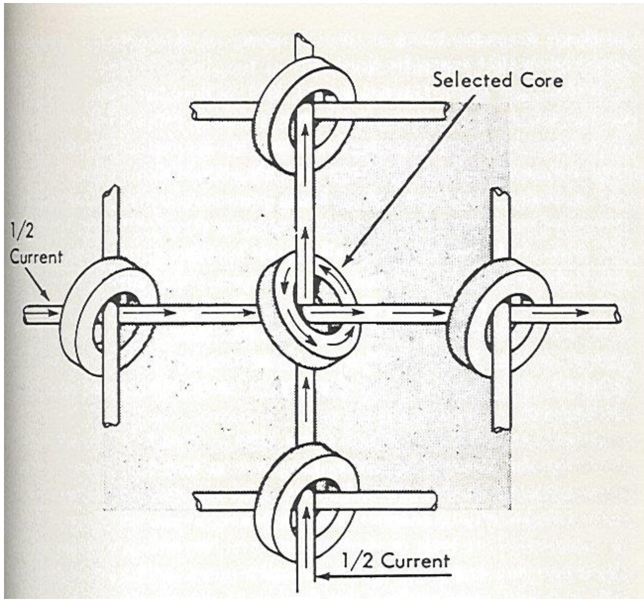
An entire work culture evolved around punched cards!

How did the IBM 1401 change all that?

IBM 1401 Innovations

- One of IBM's first all-transistor computers.
  - Earlier machines used vacuum tubes.
- Used magnetic core memory instead of a plugboard.
- A new instruction set.
- An inexpensive stored-program computer.

Magnetic Core Memory



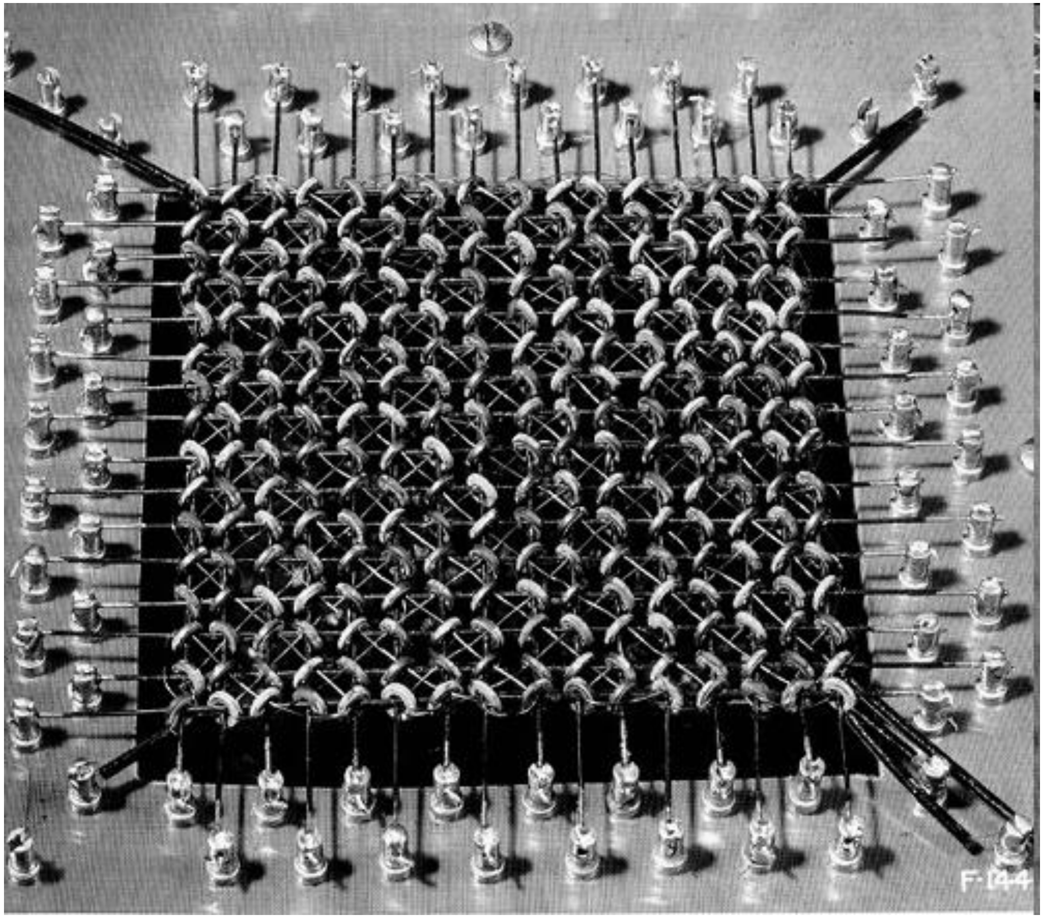
Not shown: A "sense wire" ran through each core to detect whether it was 0 or 1.

A core dump was a printout of the contents of memory.

This was the original computer usage of the word "core".

- Each bit in memory was stored by a tiny magnetized ferrite donut.
  - Either 0 or 1 depending on the magnetism direction.
  - Cores were wired together into core planes.
  - The core planes were stacked to form main memory.
  - Core memory was non-volatile.
    - Memory retained its contents even after the power was turned off.

Magnetic Core Memory, *cont'd*



A single core plane

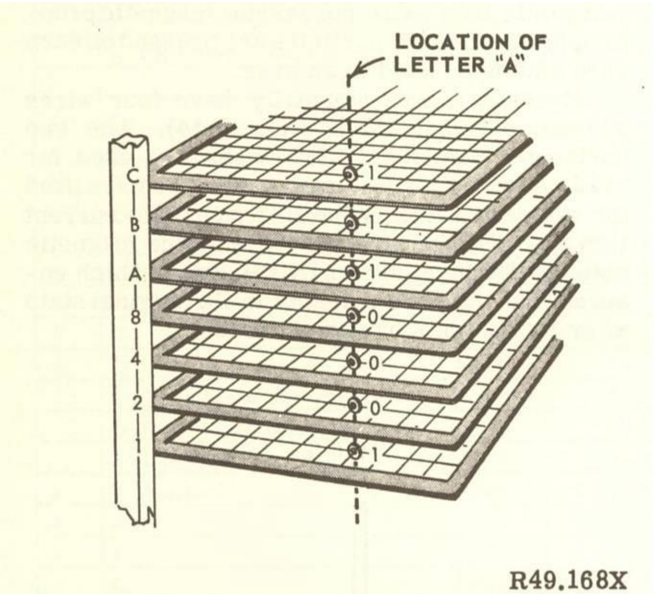


Figure 9-15.—Representing a character in core storage.

A core stack in main memory.

- In the 1401, each core bit cost 60¢ ($3/bit in today's dollars).

The IBM 1401 Computer System

- Memory was a limited resource.
  - The main CPU unit contained up to 4K characters of core memory (1 character = 8 bits).
  - You could add the IBM 1406 memory unit which contained up to 12K of additional memory
  - Maximum memory was 16K.
    - The instruction set could not address larger memory sizes.
    - You could lease smaller systems with 4K, 8K, or 12K.

K = 1000
(not 1024)

The IBM 1401 Computer System

- The 1401 computer system had amazing peripherals (I/O devices).
  - 1403 Line Printer
  - 1402 Card Reader Punch
  - 729 Magnetic Tape Drives
  - Disk drives became available later.

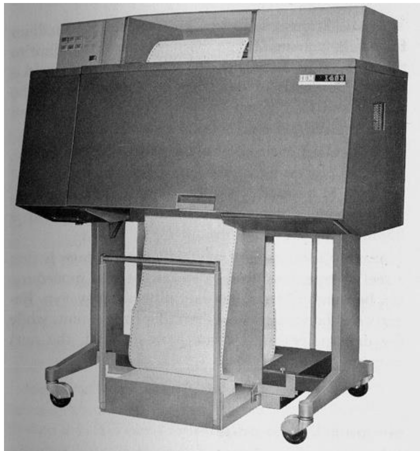


Figure 7. IBM 1403 Printer

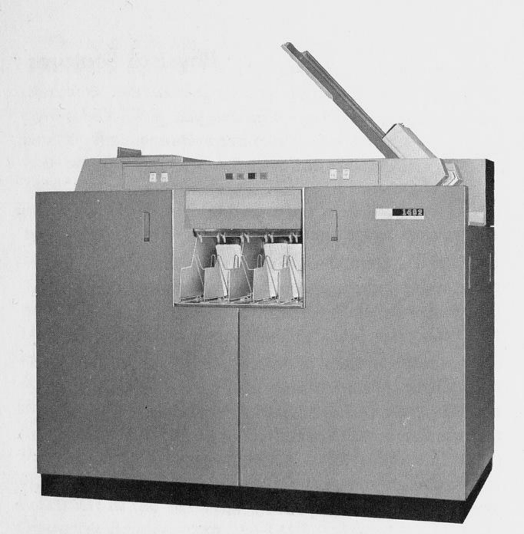
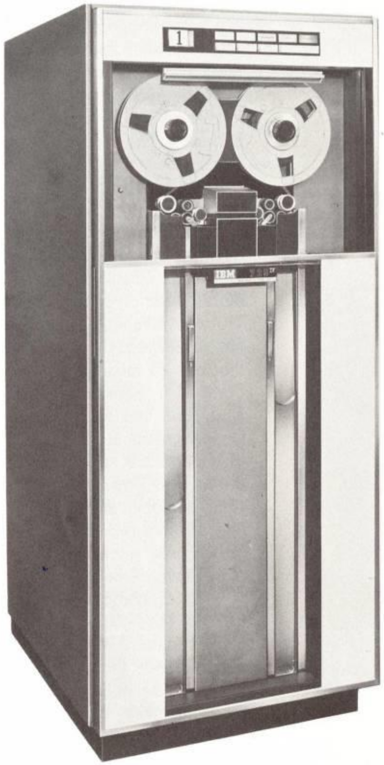


Figure 4. IBM 1402 Card Read-Punch



The 1403 Line Printer

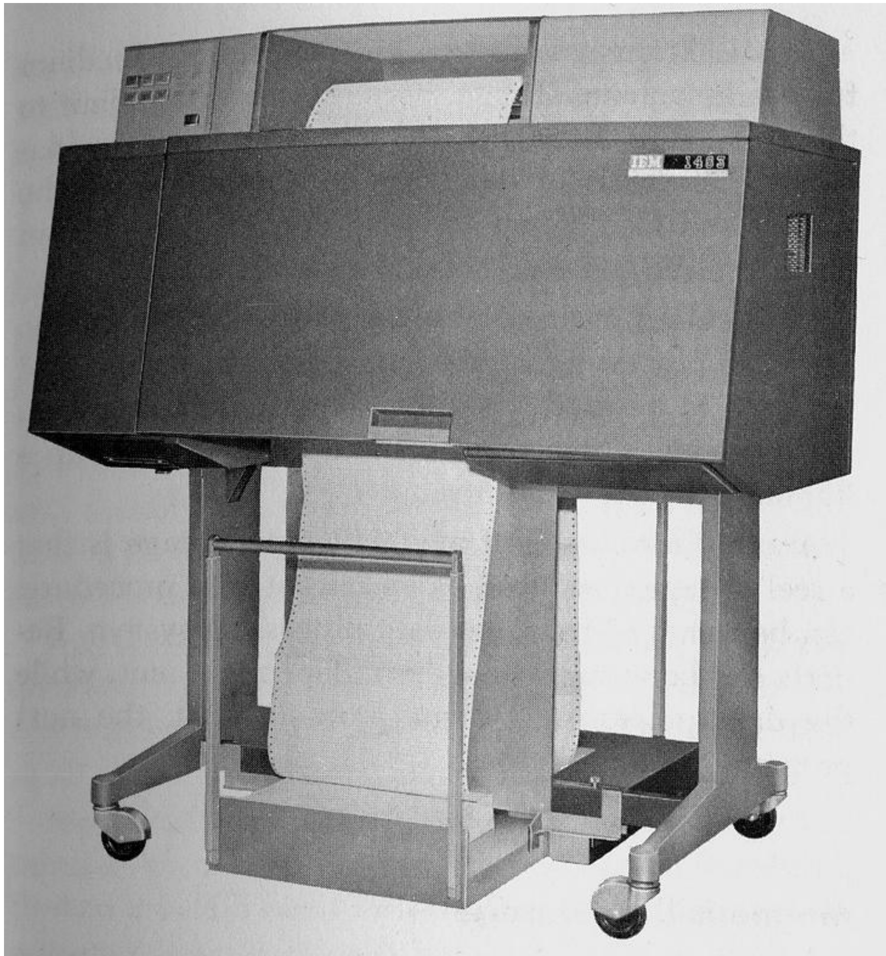


Figure 7. IBM 1403 Printer

- Each print line can contain up to 132 characters.
  - Mechanically (impact) printed.
  - No lasers!
- Outstanding print quality.
  - Horizontally straight lines of text.

Sample print quality.

The 1403 Print Mechanism

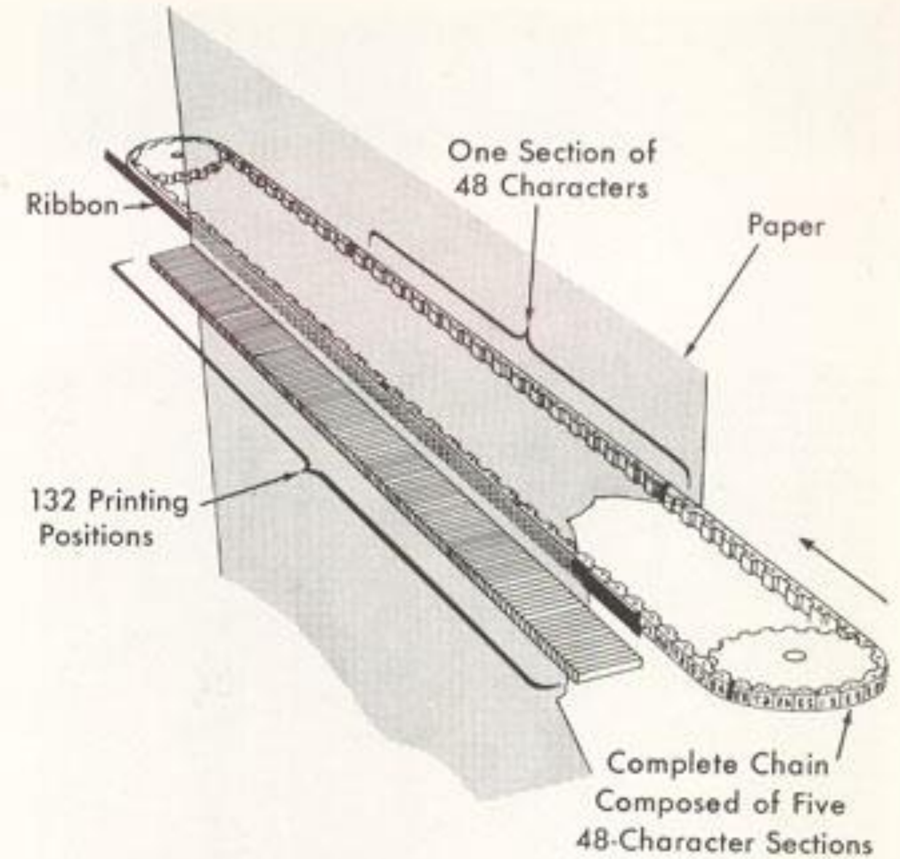
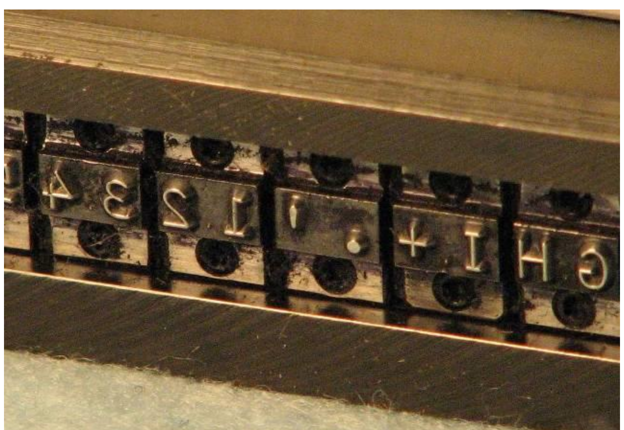


Figure 8. Printing Mechanism Schematic

- 132 horizontal print hammers behind the paper, one per print column.
  - Paper pulled upwards.
- Inked ribbon in front of the paper.
- Horizontally rotating print chain in front of the ribbon.
  - The print chain contains type slugs of the characters.
- As the desired character flies past a print column, the column's hammer fires to press the paper against the ribbon and the type slug.
  - The print chain does *not* stop.
  - The paper advances as soon as the entire line is printed.

The 1403 Print Mechanism *(cont'd)*

1403 print cartridge with print chain



Print chain magnified

- How fast was the 1403 printer?
  - <u>Up to 600 lines per minute</u>!

1402 Card Read Punch

Blank cards to be punched

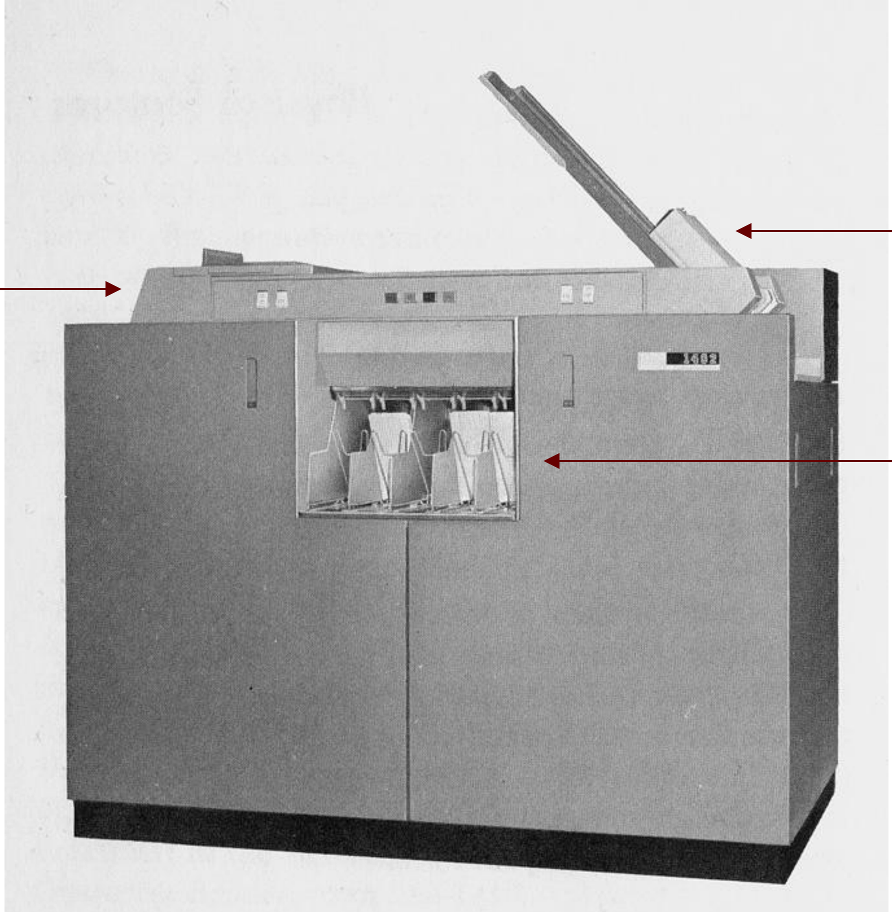


Figure 4. IBM 1402 Card Read-Punch

Stack of punched cards to be read

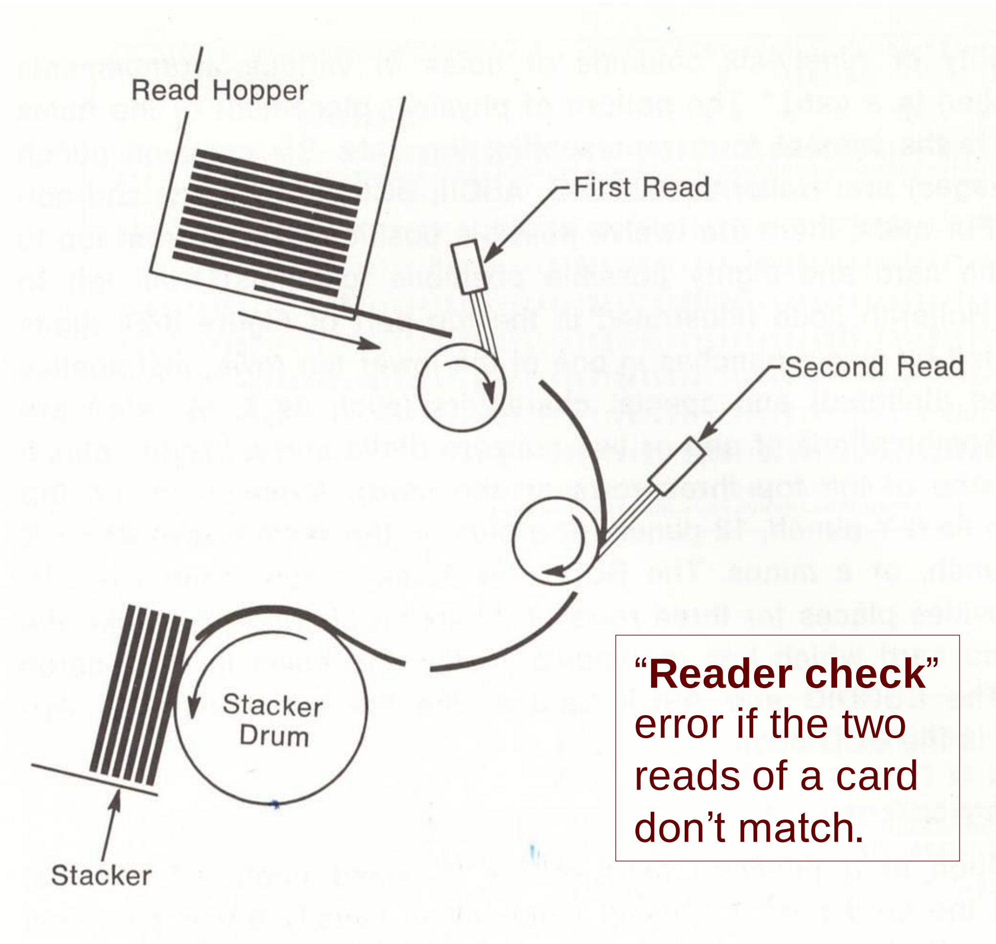
Card hoppers

1402 Card Read Punch *(cont'd)*

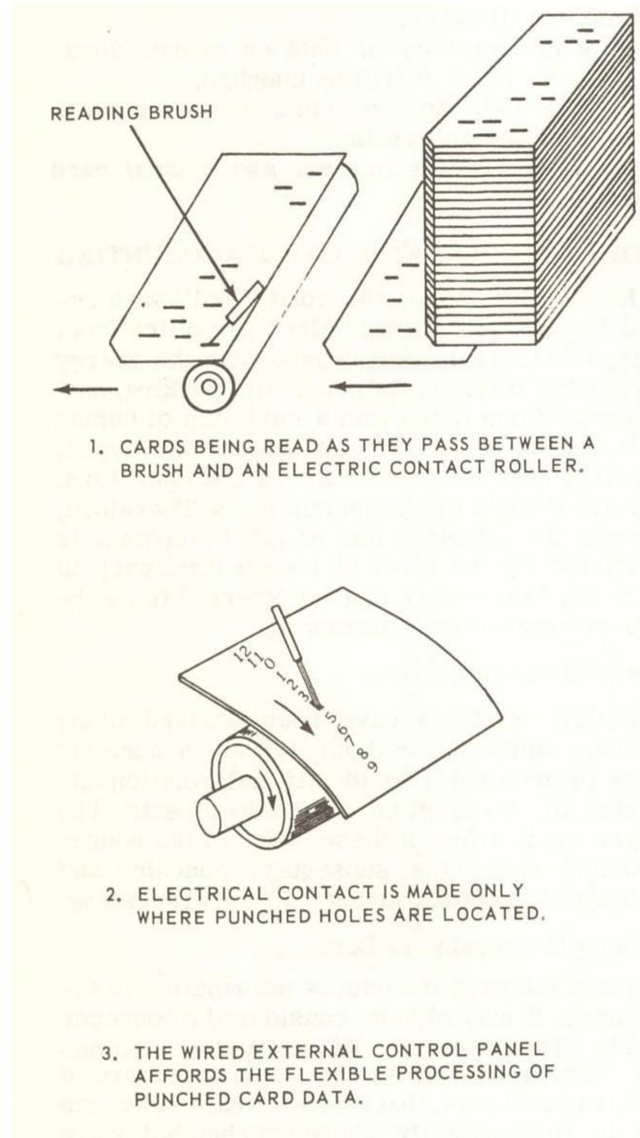
Card stacker

Punched cards

How the 1402 Read Punched Cards



"**Reader check**" error if the two reads of a card don't match.



How the 1402 Read Punched Cards *(cont'd)*

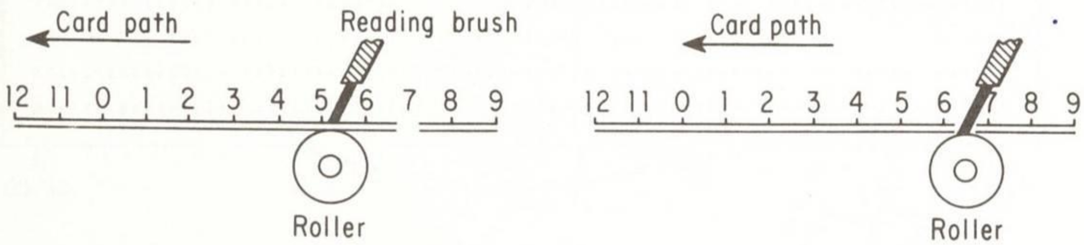
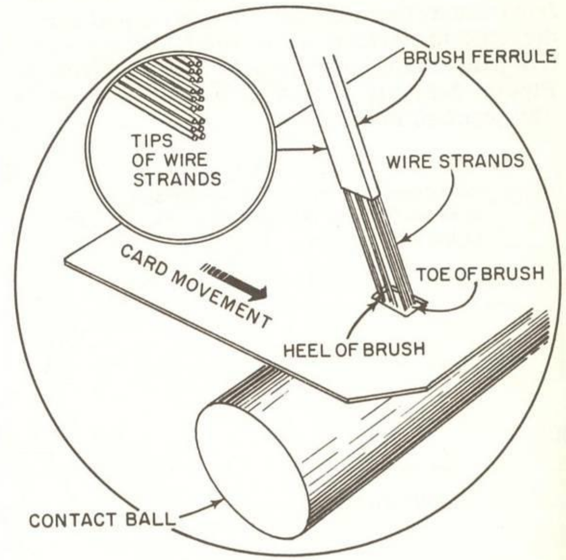
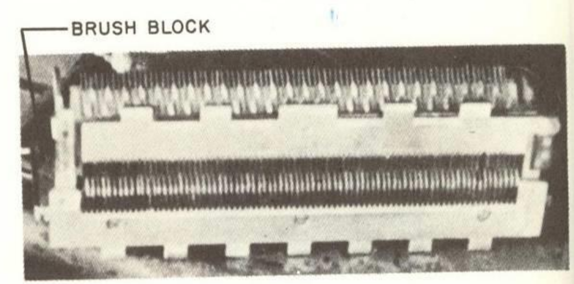


FIG. 6-3 A reading brush before (left) and during the detection of a hole in row 7 of a given column





49.229

Figure 2-28.—An 80 column brush assembly (block).

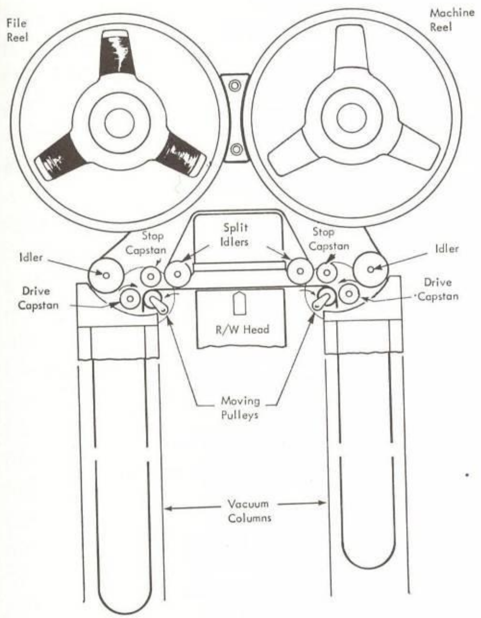
- It's all in the timing!
  - One brush per card column.
  - All 80 columns were read simultaneously.
  - Cards were fed into the 1402 card hopper "9 edge face down".
- How fast was the 1402 card reader?
  - <u>Up to 800 cards per minute</u>!

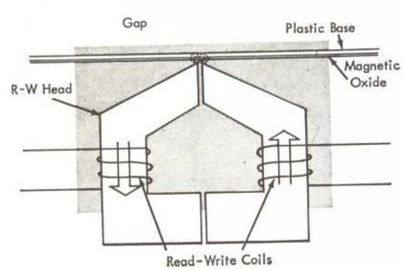
IBM 729 Magnetic Tape Drive

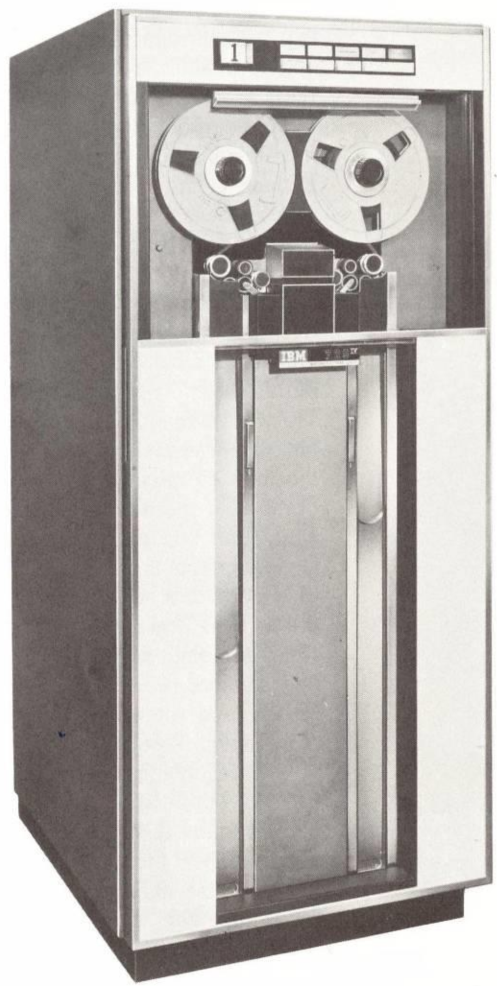
IBM 729 tape drives at NASA

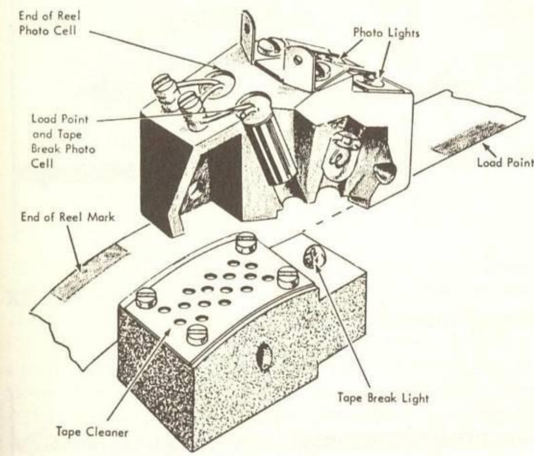
- 7-track tape
- Up to 800 characters/inch
- 480 K bits/second transfer rate
- One 2400-foot tape reel = 3 MB = 37,500 punched cards (nearly 20 boxes)
  - Tape capacity depends on block size and blocking factor.

IBM 729 Magnetic Tape Drive *(cont'd)*



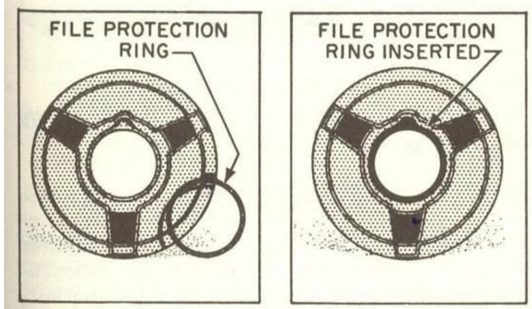






R78.30X

Figure 10-28.—Sensing of tape markers.



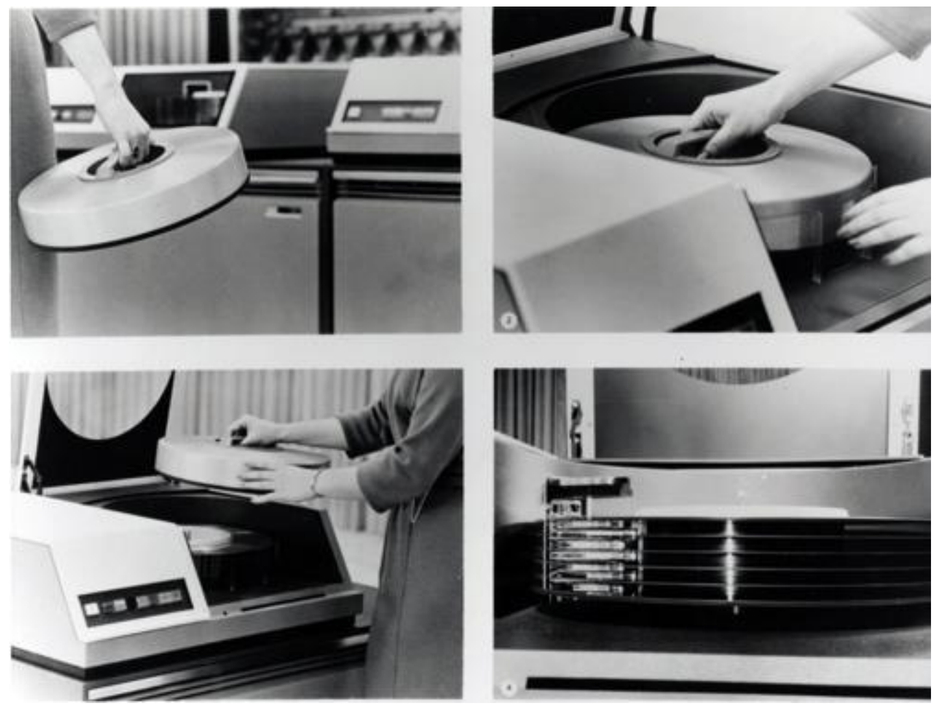
The file protection device is a plastic ring that fits into a round groove molded in the tape reel. When the ring is in place, either reading or writing can occur. When the ring is removed, writing is suppressed and only reading can take place; thus, the file is protected from accidental erasure.

R49.303X

Figure 10-29.—File protection device.

IBM 1311 Disk Drive

- Removable disk pack, each 10 lbs.
- 4-inch stack of six 14-inch disks
- 10 recording surfaces rotated at 1500 rpm
- Average <u>random-access</u> time 250 ms
- 2 MB per pack = 25,000 cards (12½ boxes)



IBM 1311 Disk Drive

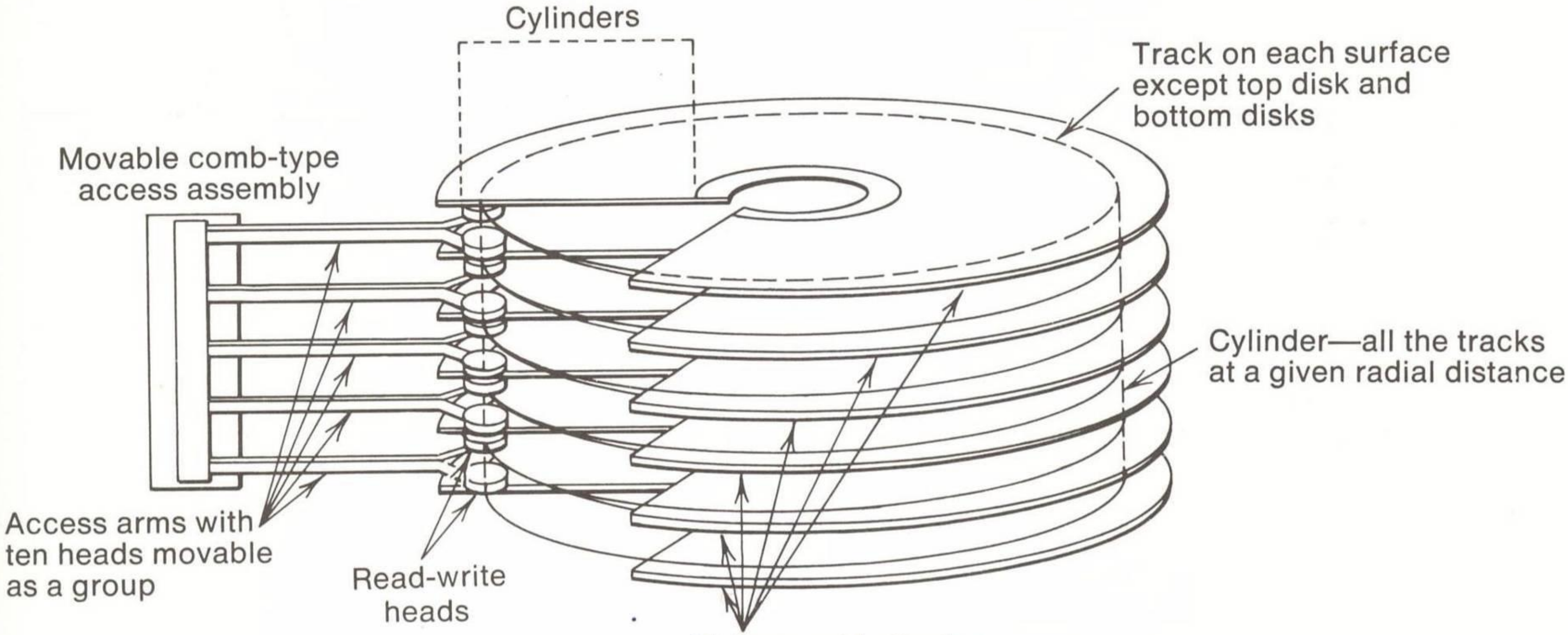


Figure 10-8. Magnetic disk terminology.

The IBM 1401 was Affordable

- Most 1401 systems were leased by businesses.
  - A typical system rented for $6500/month ($45,000 in today's dollars)
  - Purchase price was $500,000 ($3.4 million in today's dollars)
- A small or medium-sized business could afford tape and/or disk drives.
  - No more data processing using only punched cards and unit-record equipment.

The 1401 was a Huge Success!

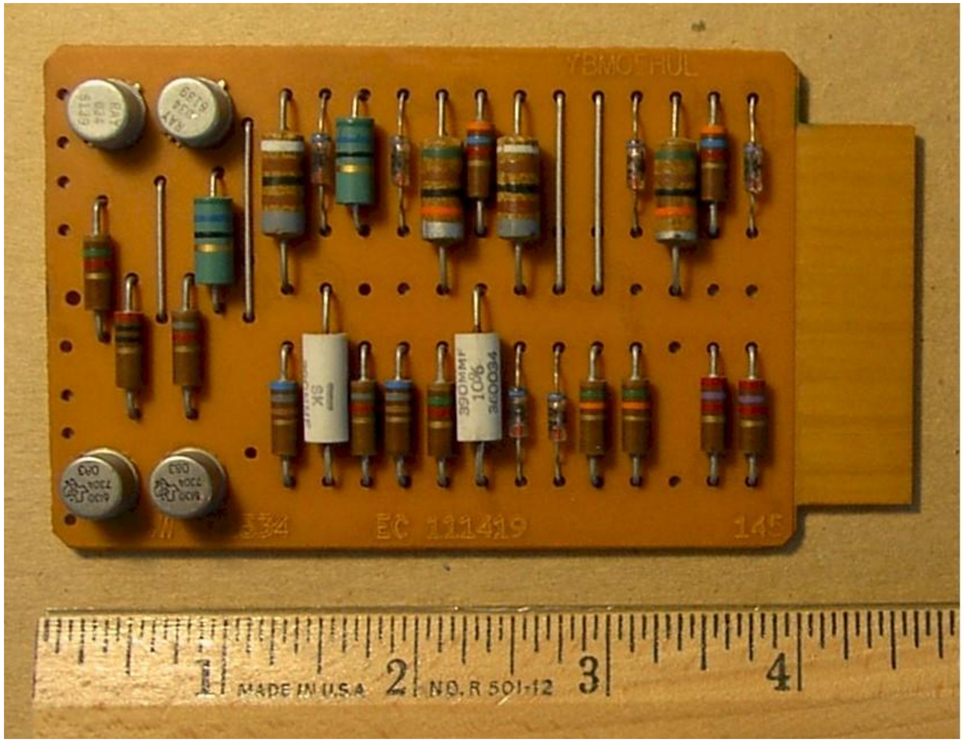
- Announced on October 5, 1959.

- 5200 systems ordered in the first 5 weeks alone.
  - More than all then-existing computers.
  - Exceeded the lifetime sales forecast.

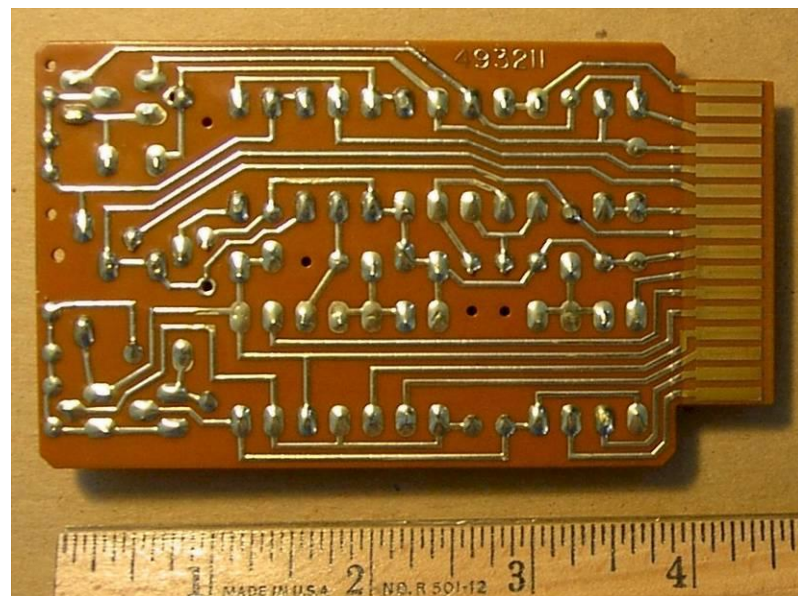
- First system delivery one year later to Time-Life, Inc.
  - Transferred 40 million punched cards to reels of magnetic tape.

- By the middle of the 1960s, half of all computers were 1401s or members of its family.
  - 1401 installations peaked at about 9300 systems.
  - All 1400 family installations peaked at about 15,600 systems.

The 1401 Architecture



- Included 3000 Standard Modular System (SMS) cards.
  - Each card contained discrete components.
  - Over 500,000 components total.
- System weighed 4 tons.
- Consumed 13,000 watts.



- Flip-flop circuit with input receivers and output drivers.
  - germanium alloy-junction transistors
  - germanium point-contact diodes
  - electrolytic capacitors
  - inductors and resistors

The 1401 Architecture

- 11.5 microsecond clock cycle:
  - Add two decimal digits
  - Move one character from one memory location to another

- 87 KHz
  - Today's 4 GHz PC can add two 20-digit numbers about 1 million times faster than a 1401.
  - A single laptop computer today has more computing power than all the 1401 systems ever installed.

The 1401 Architecture

- Each memory location contained 8 bits.
  - 6-bit character set: A-Z (upper case only), 0-9, special symbols, and control characters.
  - How the bits were labeled: **C BA 8421**

check bit (C) **zone bits** (BA) **numeric bits** (8421)

What about the 8th bit?

	C BA 8421	Punches
3	1 00 0011	3
A	0 11 0001	12-1
M	1 10 0100	11-4
S	1 01 0010	0-2
$	1 10 1011	11-3-8

- Numbers were stored as strings of decimal digits.
  - One digit per location.
  - Arithmetic was done in base 10.

For a number, the zone bits were used for the arithmetic sign or to signify an overflow.

Memory was a Limited Resource

- Up to 16K characters in memory.
  - They weren't called "bytes" back then.
- Small 1401 systems had only 4K of memory.

K = 1000

- The system designers devised ways to make the best use of the available memory locations.

Variable-Length Data

- Variable-length numbers and character strings.
  - Each piece of data used only the memory locations it needed.

- How was this accomplished?
  - The memory address pointed to the low-end (rightmost) character.
  - The 8th bit of each memory location was the word mark bit.
  - The word mark bit was on for the high-end (leftmost) character to mark the left end of the string.

- Example: Two characters strings in memory locations 400-409

```
400: HELLOWORLD
```

  - Each character that has its word mark bit set is underlined above.
  - The address of **HELLO** is 404 and the address of **WORLD** is 409.

How the 1401 Transformed Data Processing

- It was cheap enough for small businesses.
- Programs were stored in main memory.
  - No more plugboards!
- Transferred data from punched cards to magnetic tape and disk.

How the 1401 Transformed Data Processing

- Large computer centers used the 1401 as a print spooler.
  - The “big mainframe” computer wrote its output onto magnetic tapes.
  - The 1401 read the tapes and printed their contents with its high-speed 1403 printer.

How the 1401 Transformed Data Processing

- Customers could program their own machines.
  - Autocoder assembly language
  - FORTRAN, COBOL, RPG
    - The FORTRAN compiler made 63 passes and required 4K of memory.

The 1401 helped start the software industry.

How the 1401 Transformed Data Processing

- The 1401 was so popular that later IBM computers and computers from other companies would emulate its instruction set.
  - An early form of virtual machines.
- 1401 programs were running under emulation until the Y2K crisis (in the year 2000) finally killed them off.

Fully Restored IBM 1401 Systems

- The **Computer History Museum** has fully restored two complete 1401 systems.
  - You'll see, hear, and experience them in operation this afternoon during your museum visit.

1406 Memory Unit
(12 K of additional memory)

Programming the IBM 1401

- The 1401 had a simple, elegant instruction set that made it easy to program.
- There was no operating system.
- I/O operations were straightforward.
  - The **R**ead instruction read the contents of a data card directly into the "read area" at memory locations 1 – 80.
  - The **P**unch instruction punched directly to a card the contents of the "punch area" at memory locations 101 – 180.
  - The **W**rite instruction wrote directly to the printer the contents of the "print area" at memory locations 201 – 332.
  - These I/O areas were otherwise regular memory locations.

1401 Autocoder Programming

- **80/80 List**
  - Read and print a deck of cards.

```
          JOB  80/80 CARD LISTER
*
          ORG  333     LOCATE AFTER THE PRINT AREA
START     CS   332     CLEAR STORAGE 332 - 300
          CS           CLEAR STORAGE 299 - 200
          SW   1,201   SET WORD MARKS AT 1 AND 201
*
READ      R            READ A CARD INTO READ AREA
          MCW  80,280  MOVE TO PRINT AREA
          W            PRINT IT
          BLC  DONE    GO TO DONE IF LAST CARD READ
          B    READ    ELSE GO READ ANOTHER CARD
*
DONE      H    DONE    ALL DONE
          END  START
```

Main loop

MCW	Move characters to word mark

1401 Autocoder Programming

- **Powers of 2**
  - Print

```
  1
  2
  4
  8
 16
 32
 64
128
```

  - Stop when the number is 130 digits long.
  - Double the entire 130-digit value by adding it to itself each time.

Powers of 2, Version 1

```
          JOB  POWERS OF 2 VERSION 1
*
          ORG  333      LOCATE AFTER THE PRINT AREA
START     CS   332      CLEAR STORAGE 332 - 300
          CS            CLEAR STORAGE 299 - 200
          SW   203      SET WORD MARK AT MOST SIGNIFICANT DIGIT
*
          ZA   @0@,332  FILL NUMBER WITH ZEROES
          MCW  @1@,332  START WITH '1' AS LEAST SIG. DIGIT
*
LOOP      BAV  DONE     IF OVERFLOW FLAG SET THEN GO TO DONE
          W             WRITE THE NUMBER TO THE PRINTER
          A    332      ADD THE NUMBER TO ITSELF
          B    LOOP
*
DONE      H             ALL DONE
          END  START
```

Main loop

ZA	zero and add
BAV	branch if arithmetic overflow flag is set

Powers of 2, Version 2

- Add only the significant digits — not the leading zeroes.
- Don't print the leading zeroes.
- Set the word mark at the most significant (leftmost) digit to limit the size of the number.
- Move the word mark to the left one digit each time the number overflows after doubling.
- Append a new character '1' to the beginning of the number after each overflow.

Powers of 2, Version 2

```
             JOB   POWERS OF 2 VERSION 2
*
             ORG   87          INDEX REGISTER X1 (LOCATIONS 87-89)
X1           DSA   332         X1 = 332
*
             ORG   334         LOCATE AFTER THE PRINT AREA + 1
START        CS    332         CLEAR STORAGE 332 - 300
             CS                CLEAR STORAGE 299 - 200
*
MOVEWM       MZ    @0@,1&X1    CLEAR OVERFLOW BITS OF MOST SIG. DIGIT
             CW    1&X1        CLEAR OLD WORD MARK
             MCW   @1@,0&X1    APPEND '1' TO THE BEGINNING OF THE NUMBER
             SBR   X1          X1 = X1 - 1
             C     X1,@201@    IF X1 == 201
             BE    DONE        THEN ALL DONE
             SW    1&X1        ELSE SET NEW WORD MARK ONE LOC. TO THE LEFT
*
LOOP         W                 WRITE THE NUMBER TO THE PRINTER
             A     333         ADD THE NUMBER TO ITSELF
             BAV   MOVEWM      IF OVERFLOW FLAG SET THEN GO TO MOVEWM
             B     LOOP        ELSE GO TO LOOP
*
DONE         H
             END   START
```

Three index registers at memory locations 87-89 92-94 97-99

Move the word mark to the left one location and append a '1'.

Main print loop

DSA	define storage area
MZ	move zone bits
SBR	store B register

How Good a Programmer are You?

- **Compute *pi* to ?? digits**
  - Use John Machin's formula (1706):

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

  - No math library! Use the Taylor series:

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

In only 16K of memory and writing in Autocoder, how many digits of *pi* can you compute and print?

Lessons from History

- Why should you study the history of modern computing?
- What can you learn from history that you can apply now?
- “Those who cannot remember the past are condemned to repeat it .” — George Santayana

Lessons from History

- Major milestones are usually unplanned and often unrecognized until after they occur.
  - IBM itself did not realize at first how popular the 1401 would be and what effect it would have on the data processing industry.
- Not everything was invented this morning.
  - Designers 50 years ago had a few good ideas, too!
- Simple designs are the best.
  - The 1401 had a clean architecture and was easy to program.
  - It was an extremely reliable machine and simple to maintain.
- Limited resources can force the best hardware and software designers to create paradigm shifts.
  - Resources include technology, memory, speed, cost, time to market ...

For More Information

- These slides, the sample Autocoder programs, programming manuals, and the PC-based 1401 simulator and programming environment
  - http://www.cs.sjsu.edu/~mak/1401
- IBM 1401 restoration at the Computer History Museum
  - https://computerhistory.org/exhibits/ibm1401/
  - https://www.flickr.com/photos/mwichary/albums/72157604218267780/
  - https://ibm-1401.info
  - https://ibm-1401.info/new.html

How many of you will design a computer that will be restored in a museum 60 years from today and which you will then be invited to come speak about?