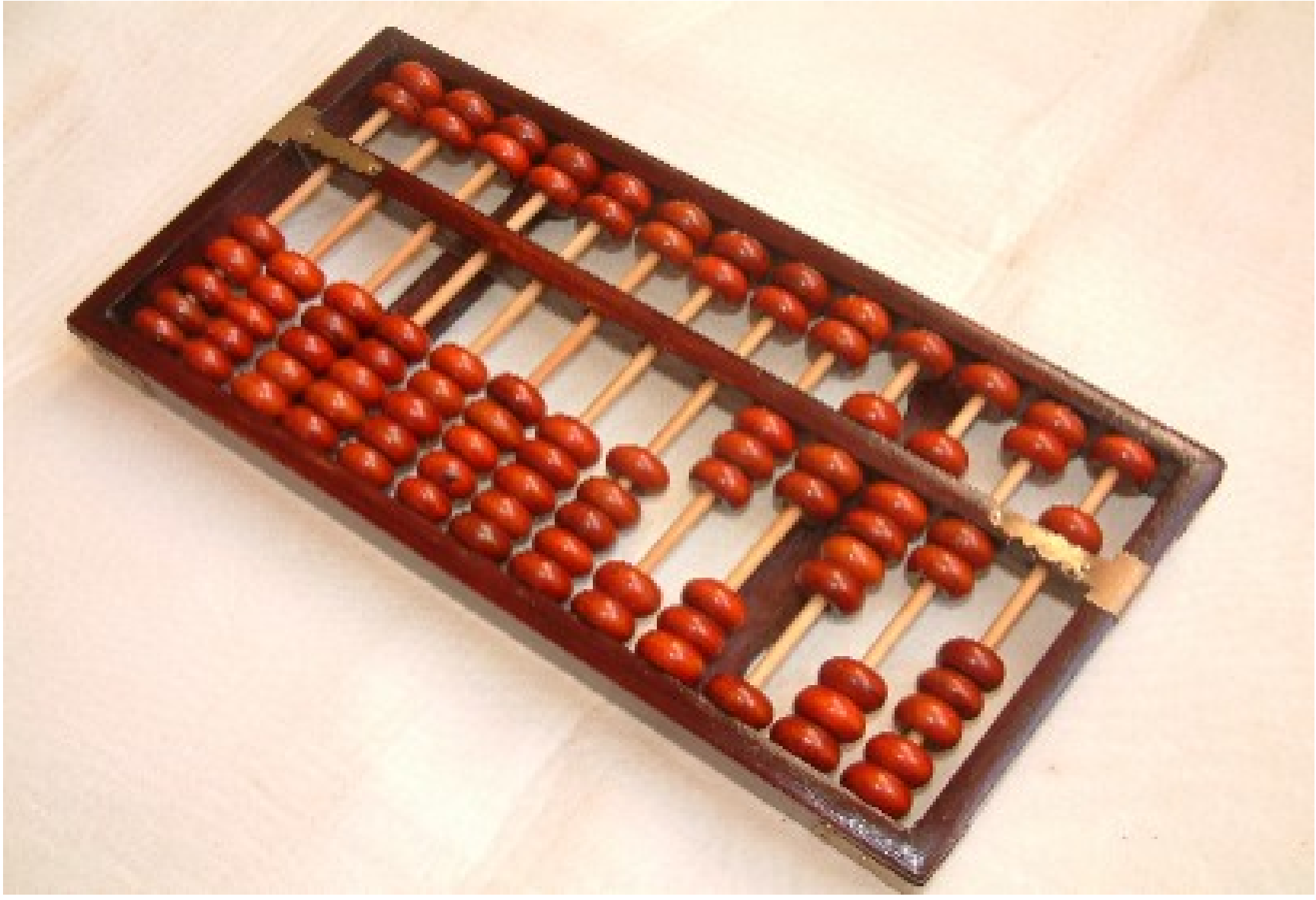
Computer Hardware: 2500 BC - wood

Abacus invented Sumeria c. 2500 BC,



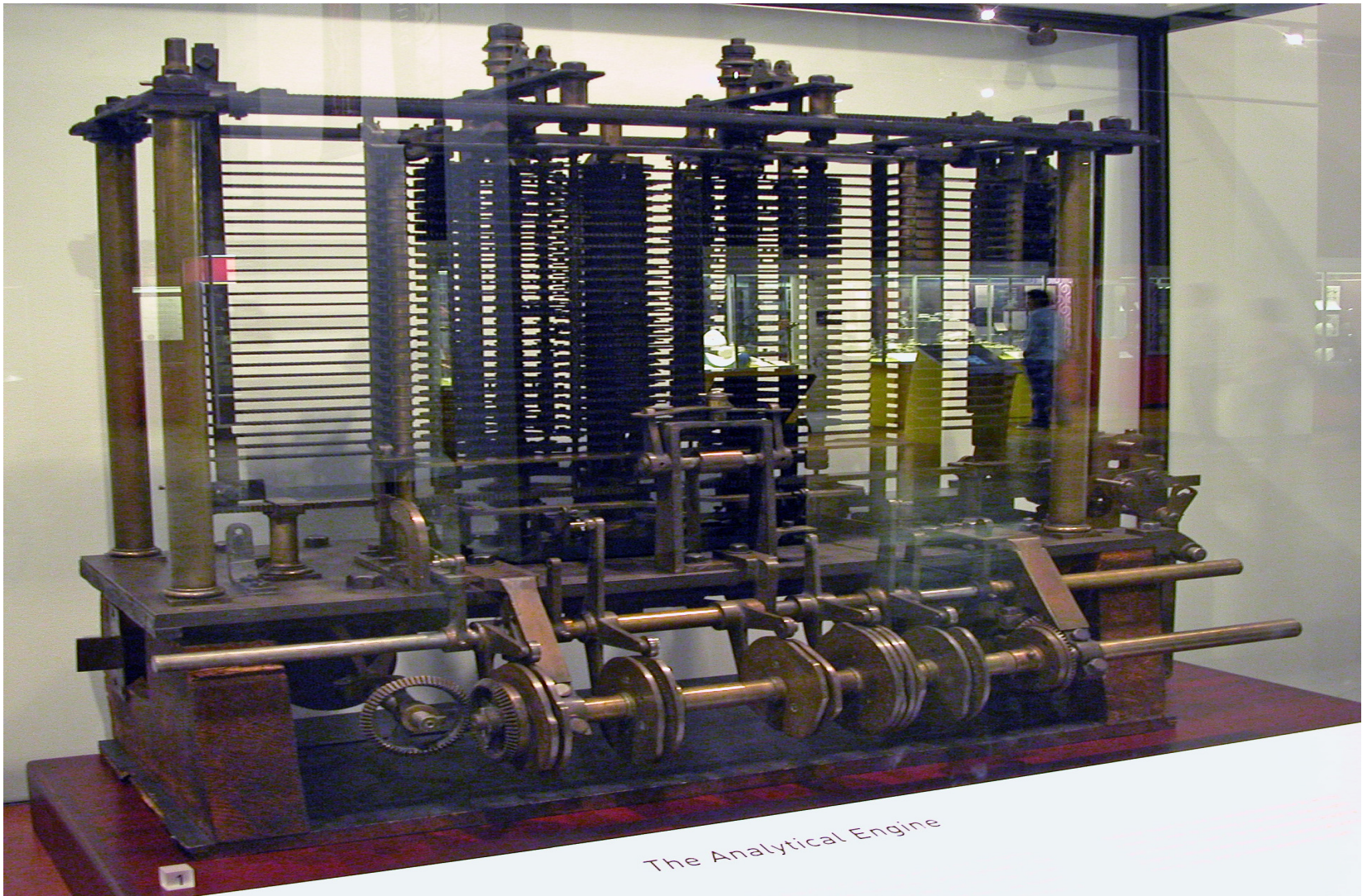
Computer Hardware: 100 BC - brass

Antikythera mechanism

Analog computer used to predict astronomical positions and eclipses

Computer Hardware: 1835 - brass & steam

Analytical Engine designed by Charles Babbage 1835 - never built. General purpose programmable computer using punch cards and steam power



The first Coder: 1835

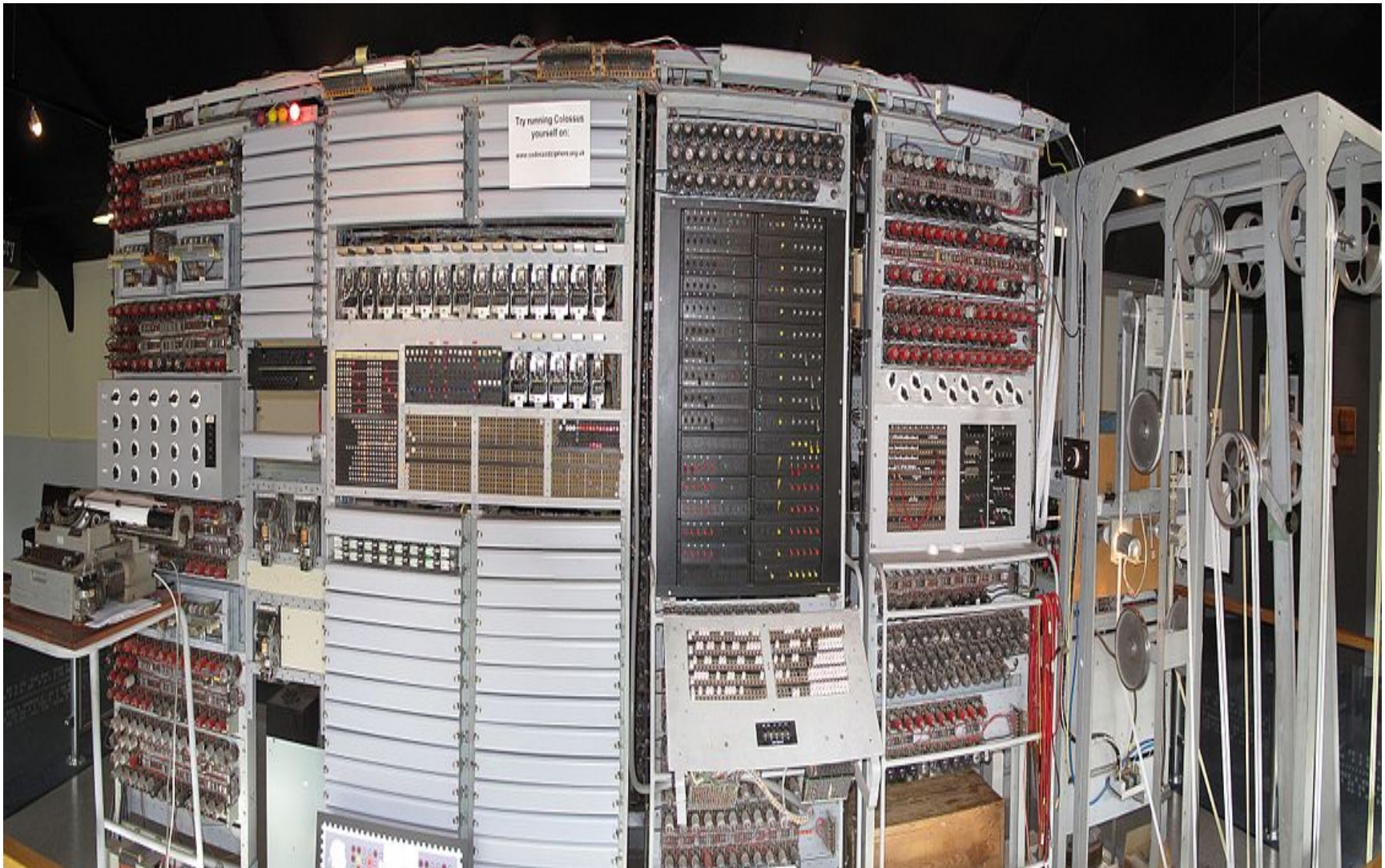
Ada Lovelace - mathematician who wrote the first programs.

Computer Hardware: 1890 - electromechanical

Hollerith tabulating machine used for calculations in the US census, company eventually becomes IBM

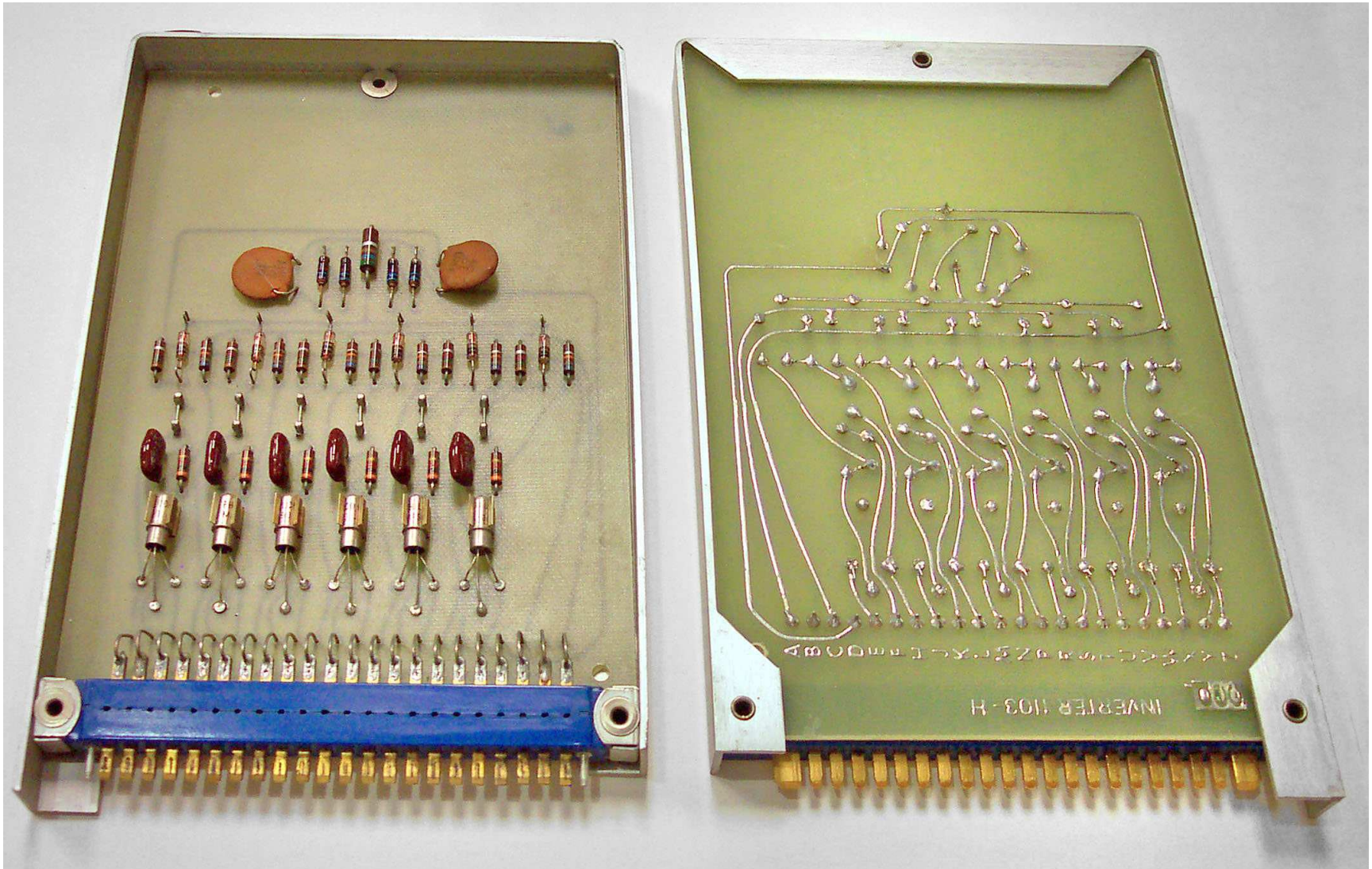
Computer Hardware: 1944 - vacuum tubes

Colossus: arguably first first programmable, electronic, digital computer.
Designed by Tommy Flowers for WWII codebreaking.



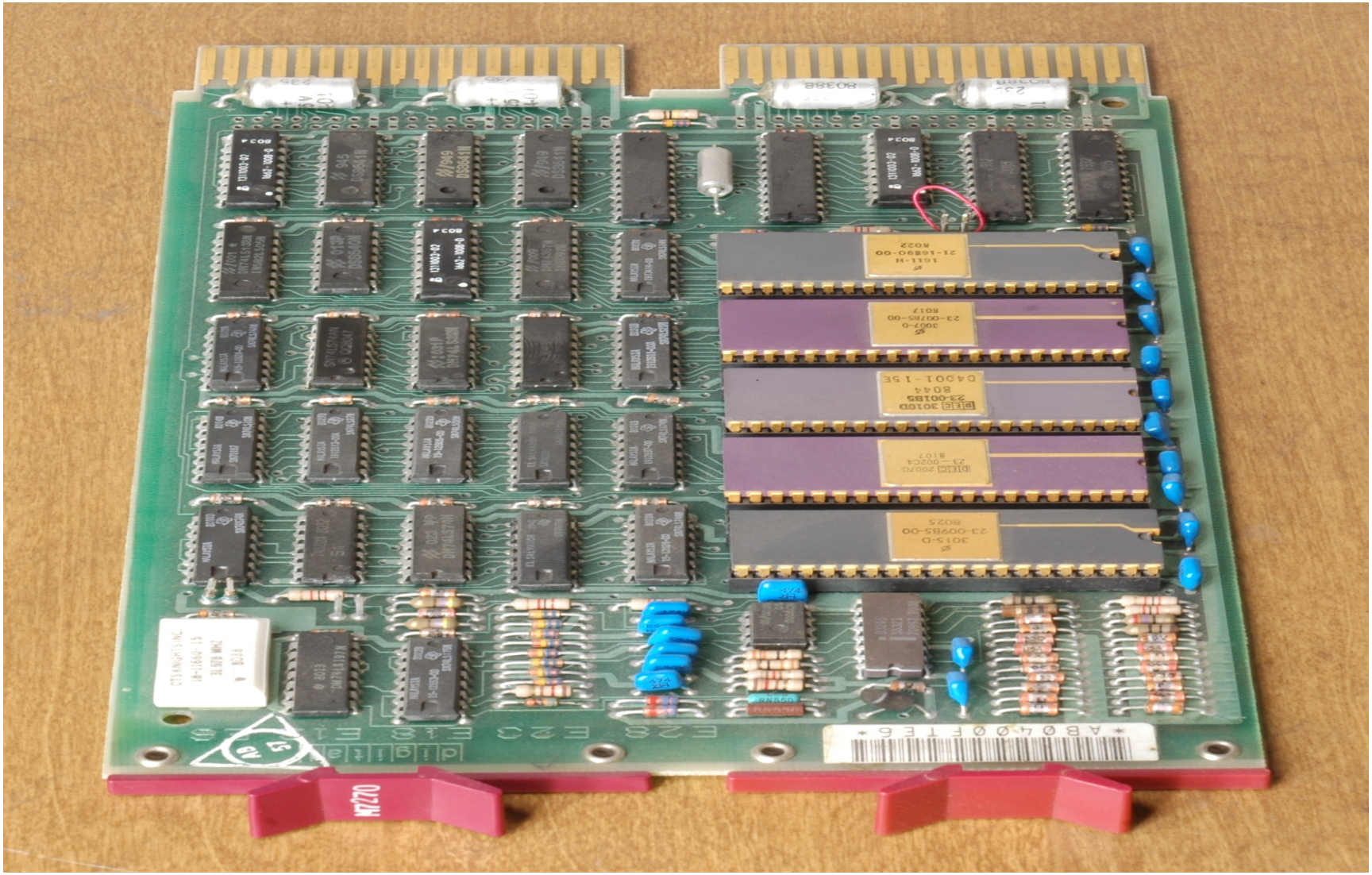
Computer Hardware: 1959 - transistors

PDP-1 first computer in Digital Equipment Corporation's successful line.
Successors were first machines C and Unix used on.



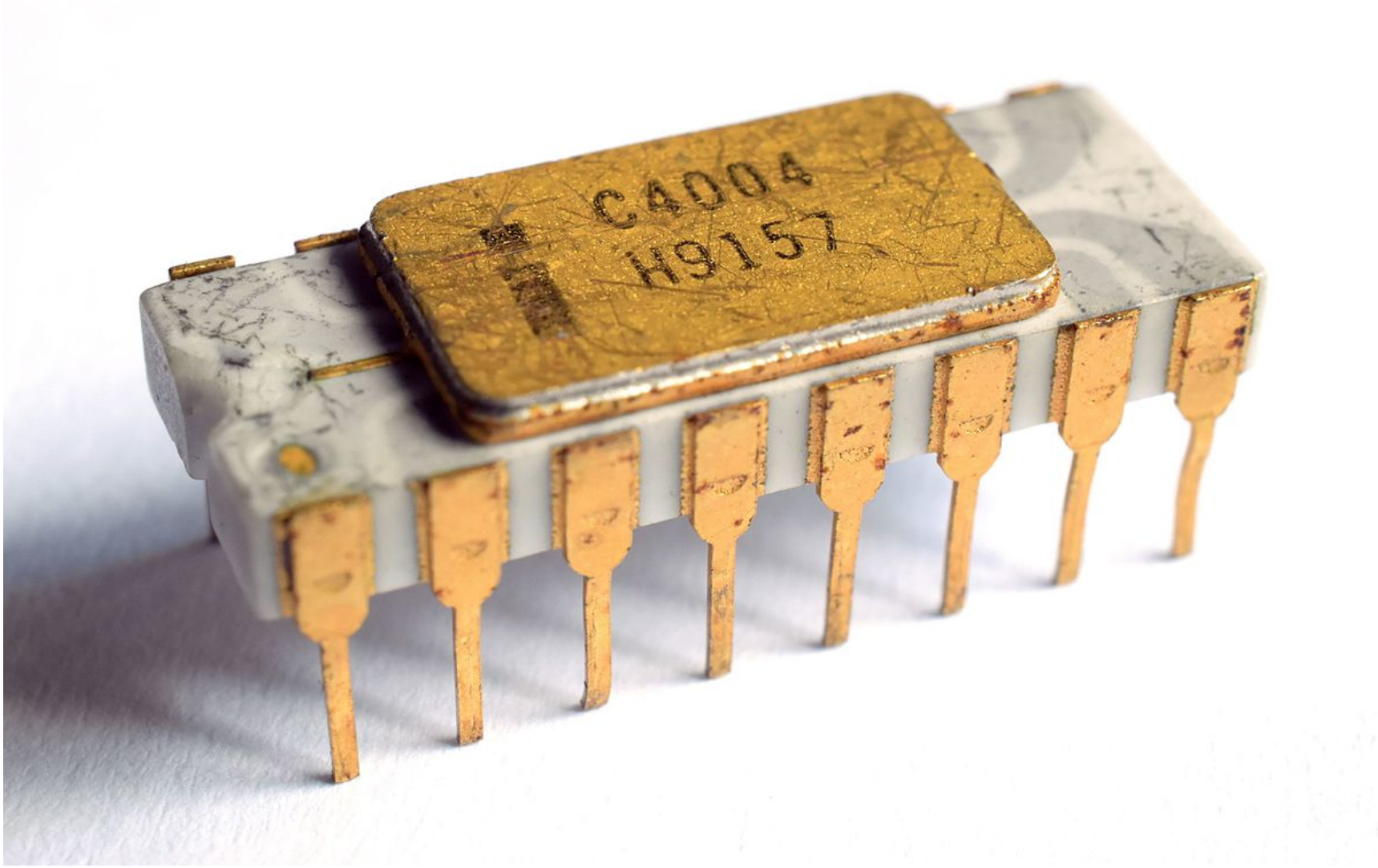
Computer Hardware: 1970 - Integrated Circuits

PDP-11 computer using large-scale integrated circuits containing thousands of transistors.



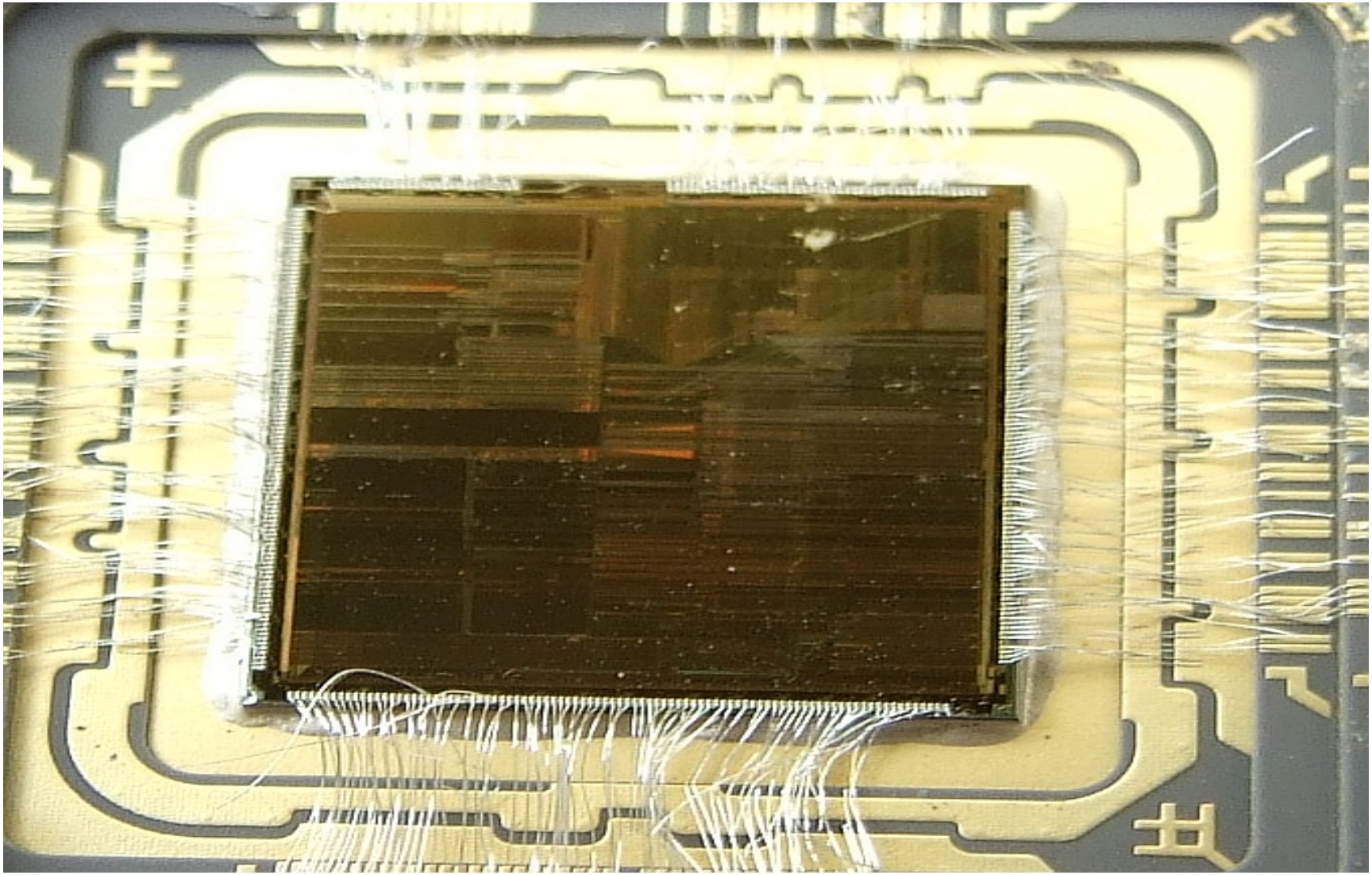
Computer Hardware: 1972 - Integrated Circuits

Intel 4004 4-bit microprocessor - computer on single chip - 2300 transistors.



Computer Hardware: 1993 - Integrated Circuits

Intel "Pentium" 32-bit microprocessor - computer on single chip - 1000000+ transistors.



The Modern Computer

What makes up a working computer?

- hardware (motherboard, CPU, RAM, HDD, etc.)
- bootstrapping code (BIOS)
- device drivers
- operating system (Linux, Windows, etc.)
- software (games, utilities, etc.)

The Operating System

Operating system (OS) is a piece of complex software layer that manages a computer's hardware.
Allows you to program without knowing (independant) of hardware details.

- examples - Windows, OS X, Linux, iOS
- long history; many innovations come from Unix
- Unix first widely used multi-user and multi-tasking OS
- Linux, Android, OSX, FreeBSD - descendants of Unix

Linux

Linux is a multi-user operating system, you will have your own account on the CSE machines, with a unique username and password. Logging in to your CSE account, either from a lab machine or from home, will give your access to your files and settings. These are not to be shared with anyone else.

- logging into a Linux system gives you access to a terminal window
- a terminal window is for text commands which the OS executes
- common commands: `ls`, `cd`, `mkdir`, etc.
- many tasks can be performed through graphical user interfaces (GUI)

Programming Languages

Why don't we program in English?

- it is too informal
- it is too big
- it is ambiguous

So we invent a programming language that:

- is small
- is formal (syntax and grammar)
- is still reasonably intuitive for humans

Because programming language instructions are usually too abstract to execute directly, they must be translated into an even more complex machine language.

The C Programming Language

Historical notes:

- created by Dennis Ritchie in the early 70's at AT&T Bell Labs
- named so because it succeeded the B programming language
- designed as a high(er)-level language to replace assembler
- powerful enough to implement the Unix kernel
- in 1978 Dennis Ritchie and Brian Kernighan published "The C Programming Language"
- now considered low-level, widely used for system and application programming

Why C?

- classic example of an imperative language
- many libraries and learning resources
- widely used for writing operating systems and compilers as well as industrial and scientifc applications
- provides low level access to machine
- language you must know if you want to work with hardware

The C Programming Language

Like most programming languages, C supports features such as:

- program comments
- declaring variables (data storage)
- assigning values to variables
- performing arithmetic operations
- performing comparison operations
- control structures, such as branching or looping
- performing input and output

Hello World

```c
// Author: Kernighan and Ritchie
// Date created: 1978
// A very simple C program.

#include <stdio.h>

int main(void) {
    printf("Hello world!\n");

    return 0;
}
```

Hello World

The program is complete, it compiles and performs a task. Even in a few lines of code there are a lot of elements:

- a comment
- a `#include` directive
- the `main` function
- a call to a library function, `printf`
- a `return` statement
- semicolons, braces and string literals

A Closer Look

What does it all mean?

- `//`, a single line comment, use `/* */` for block comments
- `#include <stdio.h>`, import the standard I/O library
- `int main(...)`, the `main` function must appear in every C program and it is the start of execution point
- `(void)`, indicating no arguments for `main`
- `printf(...)`, the usual C output function, in `stdio.h`
- `("Hello world!\n")`, argument supplied to `printf`, a *string literal*, i.e., a string constant
- `\n`, an *escape sequence*, special character combination that inserts a new line
- `return 0`, a code returned to the operating system, `0` means the program executed without error

The C Compiler

A C program must be translated into machine code to be run.
This process is known as compilation.
It is performed by a compiler.
We will use a compiler named dcc for CP1511
dcc is actually a custom wrapper around a compiler named clang.
dcc is only available on UNSW computers.
Another widely used compiler is called gcc.

Compiling A Program

- Create a file named `hello.c` containing the program
  `gedit hello.c`
- Once the code is written and saved, compile it:
  `dcc -o hello hello.c`
- Run the program:
  `./hello`

```
$ gedit hello.c &
$ dcc -o hello hello.c
$ ./hello
```

ls

- Lists files in current directory (folder)
- Several useful switches can be applied to ls
  - `ls -l` (provide a long listing)
  - `ls -a` (list all file, i.e., show hidden files)
  - `ls -t` (list files by modification time)
  - Can combine options. For example, `ls -la`

mkdir

- `mkdir` *directoryName*
- Create (make) new directory called *directoryName* in the current working directory
- a directory is like a folder in windows
- To verify creation, type `ls`

cd

- `cd` *directoryName*
- Change directory
  - Change current directory to *directoryName*
  - *directoryName* must be in the current working directory
  - We will see how to use more complex names(paths) later
- Special directory names
  - `cd ..`
    - move up one directory (to parent directory)
  - `cd`
    - move to your home directory