Computer Architecture for Everybody

From Stonehenge to Silicon

Charles R. Severance

2026

© 2026 Charles R. Severance

Contents

0.1 From Stonehenge to Silicon: Why Computers Began with Numbers

Long before computers were used for communication, entertainment, or social interaction, they were built for something far more basic: measuring the world and predicting what would happen next. The earliest forms of computation were not abstract symbols stored in memory, but physical structures designed to model natural processes. These devices helped track the seasons, predict the motion of planets, guide travelers, and support engineering and trade. In this early period, computing was inseparable from mathematics and measurement, and numbers were the primary objects being manipulated.

This focus on numerical calculation remained dominant for most of computing history. Until the late twentieth century, computers were rare, expensive, and primarily used for scientific, military, and industrial purposes. There was no internet, and few people interacted directly with computing machines. Only later did computers become tools for information exchange and personal communication. To understand modern computer architecture, it helps to begin with this earlier world, where computation meant physically transforming numbers.

0.1.1 Curved Motion and the Need for Prediction

Much of the natural world moves along curves rather than straight lines. The arc of a thrown object, the path of the Moon across the sky, and the orbit of planets all follow curved trajectories. Predicting such motion is difficult because small errors accumulate over time, and precise prediction requires repeated calculation. While the human brain is good at intuitive estimates, accurate forecasting requires systematic measurement and mathematical modeling.

The practical need to predict motion—for agriculture, navigation, and astronomy—drove the development of early computational tools. These tools were not general-purpose machines, but specialized devices that captured particular physical relationships and made them easier to reason about. Each device encoded a small piece of mathematics in wood, stone, or metal.

Seasonal sunrise and sunset alignment at Stonehenge

Figure 1: Seasonal sunrise and sunset alignment at Stonehenge

---

0.1.2 Astronomy and Physical Data Tables

From the earliest civilizations, careful observation of the sky revealed repeating patterns. The positions of the Sun, Moon, and stars changed in predictable ways over the course of days, months, and years. Recording these observations allowed calendars to be built, planting seasons to be planned, and ceremonial events to be scheduled.

Over time, physical structures were constructed to embody these patterns directly. Instead of writing numbers in tables, builders placed stones or architectural features so that sunlight or shadows would align at particular times of year. These structures functioned as durable, physical data sets that could be read simply by observing the environment.

0.1.3 Stonehenge as a Measuring Device

Stonehenge, constructed in several phases between roughly 5100 and 3600 years ago, illustrates this idea clearly. The arrangement of stones aligns with sunrise and sunset positions that change throughout the year. By observing which stones line up with the Sun on a given day, seasonal transitions can be predicted.

Rather than storing numbers, Stonehenge stored geometric relationships. Over many generations, observations were effectively "written" into the landscape. In modern terms, the structure can be thought of as a calibrated data table, built incrementally through centuries of refinement.

Similar solar and lunar alignment structures exist across many cultures, from temples in India to monuments in Central America. All reflect the same underlying idea: physical construction can encode numerical patterns from nature.

> Physical computing did not begin with machines. It began with architecture.

Antikythera mechanism gear layout

Figure 2: Antikythera mechanism gear layout

0.1.4 Continuous Representations of Number

Many early computational devices represented numbers not as symbols, but as physical positions. A value might correspond to the angle of a gear, the distance of a slider, or the rotation of a dial. Because these values could vary smoothly, such systems are called continuous or analog.

In these devices, mathematical relationships are built into geometry. Adding distances can perform multiplication when scales are logarithmic. Rotating disks can solve trigonometric problems by turning angles into lengths. Instead of executing arithmetic step by step, the device produces results through physical alignment.

Accurate printed scales are essential for this approach. The precision of the computation depends directly on the precision of the markings and the mechanical stability of the device. In effect, the manufacturing process becomes part of the calculation.

0.1.5 Gears as Models of the Solar System

The Antikythera mechanism, dating to around 2100 years ago, represents one of the most sophisticated examples of ancient analog computing. This device used interlocking gears to model the motion of the Sun, Moon, and known planets. Some of its gear trains represented long astronomical cycles spanning decades or even centuries.

Rather than calculating planetary positions numerically, the mechanism physically enacted the model of the cosmos that astronomers had developed. Turning a crank advanced time, and the gears moved accordingly. The computation occurred through mechanical interaction, not through written arithmetic.

This illustrates a broader principle that remains true today: computing systems implement models of reality. Whether those models are built from

Slide rule logarithmic scales

Figure 3: Slide rule logarithmic scales

bronze gears or silicon transistors, the purpose is the same—to predict behavior by simulating it.

0.1.6 Practical Analog Computing

Analog computation did not remain confined to astronomy. Devices such as slide rules transformed multiplication into addition by using logarithmic scales. By aligning and sliding rulers, complex calculations could be performed quickly and reliably.

A particularly practical example is the E6B flight computer, still used by pilots to compute wind correction angles and ground speed. By rotating dials and aligning scales, trigonometric relationships are solved graphically. The device does not know anything about airplanes; it simply encodes geometric laws that apply to moving vectors.

In each case, the key idea is that physical movement stands in for mathematical transformation. The device performs computation because its shape embodies mathematical relationships.

0.1.7 Discrete States and Digital Devices

Not all physical computation is continuous. Some devices operate using discrete, stable states. A mechanical latch, for example, stays in one of two positions until enough force is applied to switch it. These stable configurations allow information to be stored physically.

Clocks provide a clear example of digital behavior in mechanical systems. A pendulum provides regular timing, while ratchets and gears count discrete events. When one gear completes a full rotation, it advances the next gear, producing the familiar progression from seconds to minutes to hours.

Clock gear train with carry propagation

Figure 4: Clock gear train with carry propagation

This mechanism also introduces the concept of carry. When one digit overflows, the next digit is incremented. Mechanical systems must physically propagate this carry through connected components, a process that takes time and introduces delays. Similar effects still occur in electronic circuits, where signals must travel between components.

0.1.8 Iteration and Mechanical Automation

Once addition and counting are possible, repeated operations can be automated. Multiplication becomes repeated addition, and polynomial evaluation becomes a structured sequence of arithmetic steps. Adding motors or cranks allows machines to perform long sequences without human intervention.

Charles Babbage's Difference Engine, designed in the nineteenth century, exploited this idea. It used repeated addition to approximate complex mathematical functions and generate accurate tables. Although technology at the time could not easily produce all the required parts, a complete version built in the late twentieth century demonstrated that the design itself was sound.

> Architectural ideas often appear long before manufacturing technology can fully support them.

0.1.9 Human Computers and Early Programming

Before electronic machines became common, teams of people performed large calculations using mechanical aids and strict procedures. These workers were called computers, and their job was to execute long sequences of operations reliably.

Vacuum tube switching diagram

Figure 5: Vacuum tube switching diagram

When electronic computers emerged, much of this procedural knowledge transferred directly. Programming languages such as FORTRAN reflected existing mathematical workflows, including loops and accumulation of results. Writing a program was, in many ways, formalizing what human computers had already been doing manually.

Thus, programming did not arise as a completely new activity. It evolved naturally from structured numerical work that had existed for generations.

0.1.10 From Mechanical to Electronic Switching

Mechanical systems are limited by friction, wear, and inertia. As machines grew faster and more complex, these physical limits became obstacles. Vacuum tubes offered a way to perform switching electronically, without moving parts.

Although tubes are inherently analog devices, additional circuitry allowed them to behave digitally by latching into stable high or low voltage states. Machines such as Colossus used thousands of tubes to perform computations far faster than electromechanical systems could achieve.

Heat and power consumption remained serious challenges, but electronic switching marked a fundamental shift. Computation was no longer constrained by mechanical motion.

0.1.11 Why Architecture Begins with Numbers

Across thousands of years, computational devices were developed to support astronomy, navigation, engineering, and accounting. These systems manipulated numbers that represented physical quantities: time, distance, angle, and mass. The architectural ideas that emerged—counting, carrying, iteration, state, and switching—remain central to computer design today.

Only later did computers become tools for processing text, images, and communication. Those capabilities were built on top of numerical foundations that had already been refined through centuries of physical computing devices.

Understanding this origin helps explain why modern processors still devote most of their structure to arithmetic and control of repeated operations.

0.1.12 What Comes Next

The transition from mechanical and electronic switching to solid-state devices transformed both the speed and scale of computation. In the next chapter, attention turns to how transistors replaced tubes and gears, and how tiny electrical switches became the building blocks of modern computer systems.

0.2 From Tubes to Transistors: How Solid-State Electronics Changed Everything

The earliest electronic computers replaced mechanical motion with electrical switching, but they still relied on bulky, fragile components that consumed large amounts of power. These components, called vacuum tubes or valves, made it possible to build fully electronic machines, yet they also imposed severe limits on speed, size, and reliability. The transition from tubes to transistors did far more than improve existing designs—it reshaped what computers could be and made modern computing possible.

This chapter follows that transition. It begins with electronic amplification, moves through the physics of semiconductors, and ends with logic gates built from complementary transistor pairs. Along the way, the story shifts from analog behavior to digital abstraction, laying the groundwork for everything that follows in computer architecture.

Vacuum tube triode structure and electron flow

Figure 6: Vacuum tube triode structure and electron flow

0.2.1 Electronic Switching with Vacuum Tubes

Vacuum tubes were the first practical electronic devices capable of amplifying and switching signals. In the United Kingdom they were commonly called valves, a name that reflects their function: controlling the flow of electrons much like a valve controls the flow of water. Invented in the early twentieth century, tubes were originally developed to amplify weak analog signals, especially for long□distance telephone communication, where signals had to be boosted every few miles.

By adjusting the design of a tube, engineers could make it behave more like a switch than an amplifier. When the input voltage crossed a threshold, the output would move rapidly from low to high voltage. This made it possible to represent logical states using electrical levels: low voltage for zero and high voltage for one.

Electronic switching was dramatically faster than mechanical relays, and it had no moving parts. However, tubes required internal heaters to release electrons from the cathode, which meant high power consumption and significant heat. They were also physically large, expensive to manufacture, and prone to failure over time.

Despite these limitations, tubes enabled the first generation of fully electronic computers, including machines such as Colossus, which performed calculations at speeds that mechanical systems could not approach.

0.2.2 The Limits of Tube-Based Computing

As computers grew larger and more complex, the drawbacks of vacuum tubes became increasingly serious. Thousands of tubes meant thousands of potential failure points. Cooling systems became massive engineering projects in their own right. Electrical power consumption limited how dense and how fast circuits could become.

The fundamental problem was not the logic itself, but the physical mechanism used to implement it. A better switching device was needed—one

Crystal lattice with P-type and N-type regions

Figure 7: Crystal lattice with P-type and N-type regions

First point-contact transistor and modern MOSFET comparison

Figure 8: First point-contact transistor and modern MOSFET comparison

that did not rely on heating metal structures or maintaining vacuum environments.

That solution emerged from solid-state physics.

0.2.3 Semiconductors and Controlled Conductivity

Most materials fall into one of two categories when it comes to electricity: conductors, which allow current to flow easily, and insulators, which resist current strongly. Semiconductors occupy the middle ground. Under the right conditions, they can be made to conduct or block current in controlled ways.

Silicon, one of the most common elements in the Earth's crust, becomes useful for electronics when small amounts of other elements are added to it. This process, called doping, changes how electrons move through the crystal lattice. Regions with extra electrons are called N-type, while regions missing electrons are called P-type.

When these regions meet, electrical behavior emerges that can be precisely controlled by external voltages. At the boundaries between P-type and N-type material, electric fields form that regulate the movement of charge carriers. These microscopic interactions make it possible to build devices that switch and amplify signals without moving parts or heaters.

0.2.4 Transistors as Electronic Switches

The transistor was developed as a solid-state replacement for the triode vacuum tube. Like tubes, transistors could amplify signals and could also be

designed to behave as switches. But unlike tubes, transistors were small, required little power, and generated far less heat.

Early transistors were built using bipolar junction designs, where current flowing through one region controlled current in another. Later designs used metal-oxide-semiconductor structures, where an electric field at a control terminal called the gate regulated conduction through a channel of semiconductor material.

In both cases, the key property was the same: a small input signal could reliably control a larger output current. This allowed transistors to serve as building blocks for both analog amplifiers and digital logic circuits.

As manufacturing techniques improved, transistors became smaller, faster, and cheaper. What began as laboratory prototypes soon became mass-produced components found in radios, calculators, and eventually computers.

0.2.5 Analog Amplifiers and Digital Switching

Transistors did not immediately replace tubes in all applications. Audio amplification, for example, has long relied on both technologies, and some musicians and audio engineers still prefer tube-based amplifiers for their characteristic distortion patterns.

However, digital logic places very different demands on electronic components. In digital systems, what matters most is not the precise shape of a waveform, but whether a signal is interpreted as high or low. Designs that move quickly and decisively between these two states are far more useful than those that reproduce subtle analog variations.

To support digital operation, transistor designs were tuned to behave like fast, reliable switches rather than smooth amplifiers. Circuits were engineered so that small deviations in input voltage would be corrected at the output, restoring clean logical values. This behavior, called signal regeneration, is essential for building large digital systems where noise and small errors would otherwise accumulate.

NMOS and PMOS transistor symbols and conduction paths

Figure 9: NMOS and PMOS transistor symbols and conduction paths

CMOS inverter layout showing pull-up and pull-down networks

Figure 10: CMOS inverter layout showing pull-up and pull-down networks

0.2.6 NMOS, PMOS, and Complementary Pairs

Different transistor structures conduct under different conditions. NMOS transistors conduct when their gate voltage is high, while PMOS transistors conduct when their gate voltage is low. Each type has advantages and disadvantages when used alone.

Early integrated circuits often used only one type of transistor, resulting in designs that consumed power even when not switching and generated significant heat. As chip densities increased, this became a serious limitation.

The solution was to pair NMOS and PMOS transistors in complementary configurations. In such arrangements, when one transistor is on, the other is off. This drastically reduces static power consumption and improves switching behavior. The resulting technology is called CMOS, short for Complementary Metal□Oxide□Semiconductor.

Once CMOS manufacturing became practical in the late twentieth century, it transformed computer design. Entire processors could be placed on single chips, power requirements dropped dramatically, and circuit densities increased by orders of magnitude.

From this point forward, nearly all digital logic in mainstream computing has been built using CMOS technology.

0.2.7 From Transistors to Logic Gates

While circuits are built from transistors, designers rarely think in terms of individual switching devices when creating complex systems. Instead, transistors are grouped into higher-level structures called logic gates. A gate accepts one or more binary inputs and produces a binary output according to a logical rule.

NAND and NOR gate transistor-level structures

Figure 11: NAND and NOR gate transistor-level structures

The simplest gate is the inverter, or NOT gate, which produces the opposite of its input. In CMOS, an inverter is built from one NMOS transistor and one PMOS transistor arranged so that exactly one conducts at any time. When the input is low, the PMOS transistor pulls the output high. When the input is high, the NMOS transistor pulls the output low.

This complementary behavior provides fast switching and low power usage, making it ideal for large-scale digital systems.

0.2.8 Building Complexity from Simple Gates

Once reliable gates are available, more complex logical functions can be constructed by combining them. Gates such as AND, OR, and exclusive OR implement familiar logical operations. Other gates, such as NAND and NOR, are especially important because entire digital systems can be built using only one of these gate types.

Designing digital circuits at the gate level allows engineers to reason about behavior using binary logic instead of voltage levels and transistor physics. This abstraction makes it possible to build systems containing billions of transistors without managing each device individually.

From half adders and full adders to registers and processors, nearly every component in a computer can be described as an arrangement of logic gates operating in synchronized patterns.

0.2.9 Why CMOS Made Modern Computers Possible

Before CMOS, logic circuits were relatively slow, generated large amounts of heat, and could not be densely packed. Computers filled rooms and required elaborate cooling and power infrastructure. With the rise of CMOS, these constraints relaxed dramatically.

As transistor sizes shrank and manufacturing improved, entire central processing units moved from cabinets to circuit boards, and then onto single integrated chips. Power efficiency improved, clock speeds increased, and reliability soared.

This shift did not change the logical structure of computation, but it changed the physical feasibility of building large and fast systems. The same architectural ideas—state, control, and iteration—could now be implemented at scales that earlier engineers could only imagine.

0.2.10 Summary: Solid-State Foundations of Digital Logic

The move from vacuum tubes to transistors replaced fragile, power-hungry components with small, efficient solid-state devices. Advances in semiconductor physics and manufacturing enabled reliable electronic switching without mechanical motion or heated filaments.

Complementary transistor designs made CMOS the dominant technology for digital logic, allowing dense, low-power circuits to become practical. By grouping transistors into logic gates, designers gained an abstraction that supports the construction of complex systems without constant reference to underlying physics.

With these foundations in place, attention can now shift from individual devices to how large collections of gates are organized into functional computing units.

0.2.11 What Comes Next

With solid-state logic established, the next step is to explore how gates are combined into arithmetic circuits and memory structures. The following chapter examines how simple logical components are assembled into adders, registers, and the building blocks of processors.