



howstuffworks

home Search go

ComputerStuff AutoStuff ElectronicsStuff ScienceStuff HomeStuff EntertainmentStuff MoneyStuff TravelStuff PeopleStuff

Categories

- > Hardware
- > Internet
- > Peripherals
- > Security
- > ShortStuff
- > Software
- > Browse the Computer Library

Top Subjects

- > CD Burners
- > Hard Disks
- > Home Networking
- > Lock Picking
- > Web Servers

Sponsored By:

Explore Stuff

- > Big List of Articles
- > Get the Newsletter
- > Search HSW & the Web
- > Shop or Compare Prices
- > Orbitz - Plan a Trip!
- > Browse the Classifieds
- > Forums - Share Your Thoughts!

[Main](#) > [Computer](#) > [Hardware](#)

How Microprocessors Work

by [Marshall Brain](#)



- > [Introduction to How Microprocessors Work](#)
- > [Microprocessor History](#)
- > [Inside a Microprocessor](#)
- > [RAM and ROM](#)
- > [Microprocessor Instructions](#)
- > [Microprocessor Performance](#)
- > [Lots More Information!](#)
- > [Shop or Compare Prices](#)

Inside a Microprocessor

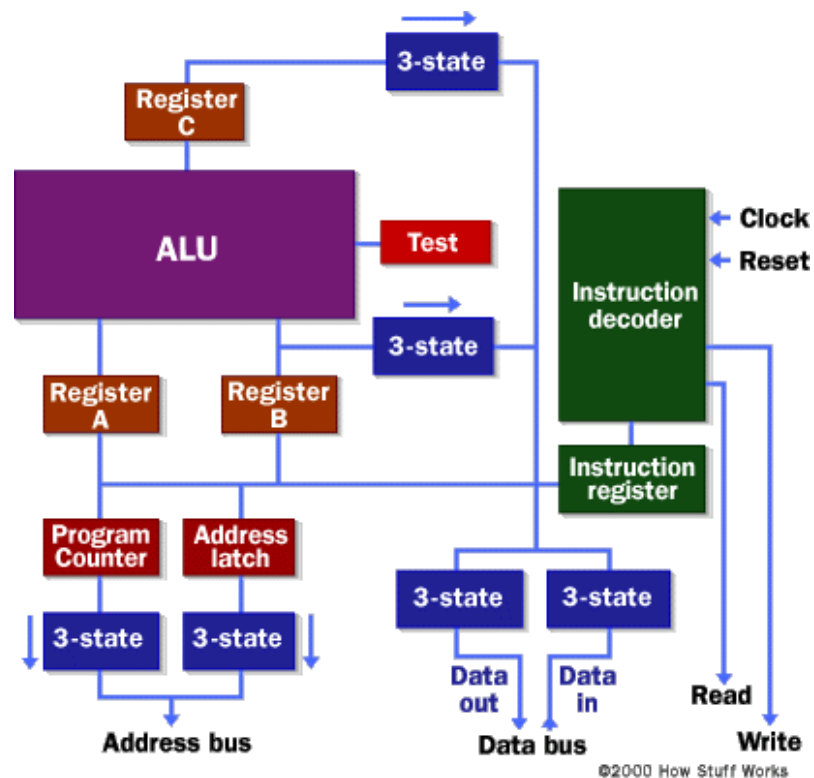
To understand how a microprocessor works, it is helpful to look inside and learn about the logic used to create one. In the process you can also learn about **assembly language** -- the native language of a microprocessor -- and many of the things that engineers can do to boost the speed of a processor.

A microprocessor executes a collection of machine instructions that tell the processor what to do. Based on the instructions, a microprocessor does three basic things:

- Using its ALU (Arithmetic/Logic Unit), a microprocessor can perform mathematical operations like addition, subtraction, multiplication and division. Modern microprocessors contain complete floating point processors that can perform extremely sophisticated operations on large floating point numbers.
- A microprocessor can move data from one [memory](#) location to another.
- A microprocessor can make decisions and jump to a new set of instructions based on those decisions.

There may be very sophisticated things that a microprocessor does, but those are its three basic activities. The following diagram shows an extremely simple microprocessor capable

of doing those three things:



This is about as simple as a microprocessor gets. This microprocessor has:

- An **address bus** (that may be 8, 16 or 32 bits wide) that sends an address to memory
- A **data bus** (that may be 8, 16 or 32 bits wide) that can send data to memory or receive data from memory
- An **RD** (read) and **WR** (write) line to tell the memory whether it wants to set or get the addressed location
- A **clock line** that lets a clock pulse sequence the processor
- A **reset line** that resets the program counter to zero (or whatever) and restarts execution

Let's assume that both the address and data buses are 8 bits wide in this example.

Here are the components of this simple microprocessor:

- Registers A, B and C are simply latches made out of flip-flops. (See the section on "edge-triggered latches" in [How Boolean Logic Works](#) for details.)
- The address latch is just like registers A, B and C.
- The program counter is a latch with the extra ability to increment by 1 when told to do so, and also to reset to zero when told to do so.
- The ALU could be as simple as an 8-bit adder (see the section on adders in [How Boolean Logic Works](#) for details), or it might be able to add, subtract, multiply and divide 8-bit values. Let's assume the latter here.
- The test register is a special latch that can hold values from comparisons performed in the ALU. An ALU can normally compare two numbers and determine if they are equal, if one is greater than the other, etc. The test register can also normally hold a carry bit from the last stage of the adder. It stores these values in flip-flops and then the instruction decoder can use the values to make decisions.
- There are six boxes marked "3-State" in the diagram. These are **tri-state buffers**. A tri-state buffer can pass a 1, a 0 or it can essentially disconnect its output (imagine a switch that totally disconnects the output line from the wire that the output is heading toward). A tri-state buffer allows multiple outputs to connect to a wire, but only one of them to actually drive a 1 or a 0 onto the line.
- The instruction register and instruction decoder are responsible for controlling all of the other components.

Although they are not shown in this diagram, there would be control lines from the instruction decoder that would:

- Tell the A register to latch the value currently on the data bus
- Tell the B register to latch the value currently on the data bus
- Tell the C register to latch the value currently on the data bus
- Tell the program counter register to latch the value currently on the data bus
- Tell the address register to latch the value currently on the data bus
- Tell the instruction register to latch the value currently on the data bus
- Tell the program counter to increment
- Tell the program counter to reset to zero
- Activate any of the six tri-state buffers (six separate lines)
- Tell the ALU what operation to perform
- Tell the test register to latch the ALU's test bits
- Activate the RD line
- Activate the WR line

Helpful Articles

If you are new to digital logic, you may find the following articles helpful in understanding this section:

- [How Bytes and Bits Work](#)
- [How Boolean Logic Works](#)
- [How Electronic Gates Work](#)

Coming into the instruction decoder are the bits from the test register and the clock line, as well as the bits from the instruction register.

<< [Prev Page](#) [Intro](#) [Next Page](#) >>

[Home](#) [HSW Home](#)

Table of Contents:

- > [Introduction to How Microprocessors Work](#)
- > [Microprocessor History](#)
- > [Inside a Microprocessor](#)
- > [RAM and ROM](#)
- > [Microprocessor Instructions](#)
- > [Microprocessor Performance](#)
- > [Lots More Information!](#)
- > [Shop or Compare Prices](#)

[Rate this Article!](#)