# But how does a Computer work? The ALU

Jaden Furtado

## Abstract

**In the previous paper, we explored how a CPU works on a high level. In this paper, we will do a deep dive into how an ALU works.**

## 1. The ALU

The ALU, or arithmetic logic unit, is the component of the CPU that is used to perform mathematical and logical operations on the inputs provided to it.

"This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. It is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs)"[1]

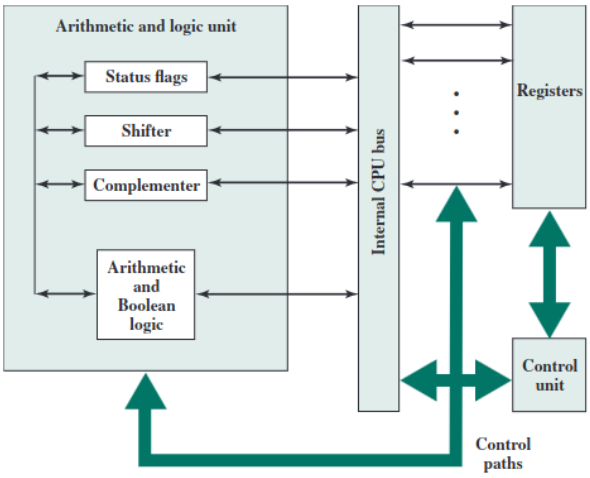


Fig 1. Internal CPU architecture

In essence, all other components bring data to the ALU and take the results of operations that are performed by the ALU.[2]

## 2. The internal circuitry

The below is the combinational logic circuitry of the 74181 integrated circuit, an early four-bit ALU, with logic gates

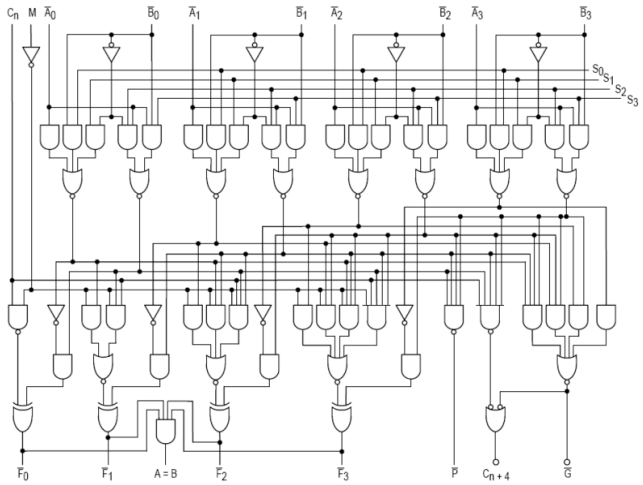

Fig 2. The combinational logic circuitry of the 74181 integrated circuit, an early four-bit ALU, with logic gates [1]

## 3. Signals

An ALU has a variety of input and output nets, which are the electrical conductors used to convey digital signals between the ALU and external circuitry. When an ALU is operating, external circuits apply signals to the ALU inputs and, in response, the ALU produces and conveys signals to external circuitry via its outputs.

### 3.1 Data

A basic ALU has three parallel data buses consisting of two input operands (A and B) and a result output (Y). Each data bus is a group of signals that conveys one binary integer number. Typically, the A, B and Y bus widths (the number of signals comprising each bus) are identical and match the native word size of the external circuitry (e.g., the encapsulating CPU or other processor).

### 3.2 Opcode

The opcode input is a parallel bus that conveys to the ALU an operation selection code, which is an enumerated value that specifies the desired arithmetic or logic operation to be performed by the ALU. The opcode size (its bus width) determines the maximum number of distinct operations the ALU can perform; for example, a four-bit opcode can specify up to sixteen different ALU operations. Generally, an ALU opcode is not the same as a machine language opcode, though in some cases it may be directly encoded as a bit field within a machine language opcode.

### 3.3 Status

[1] https://en.wikipedia.org/wiki/Arithmetic_logic_unit

**Outputs:** The status outputs are various individual signals that convey supplemental information about the result of the current ALU operation. General-purpose ALUs commonly have status signals such as:

- Carry-out, which conveys the carry resulting from an addition operation, the borrow resulting from a subtraction operation, or the overflow bit resulting from a binary shift operation.
- Zero, which indicates all bits of Y are logic zero.
- Negative, which indicates the result of an arithmetic operation is negative.
- Overflow, which indicates the result of an arithmetic operation has exceeded the numeric range of Y.
- Parity, which indicates whether an even or odd number of bits in Y are logic one.

Upon completion of each ALU operation, the status output signals are usually stored in external registers to make them available for future ALU operations (e.g., to implement multiple-precision arithmetic) or for controlling conditional branching. The collection of bit registers that store the status outputs are often treated as a single, multi-bit register, which is referred to as the "status register" or "condition code register".

### 3.4 Inputs

The status inputs allow additional information to be made available to the ALU when performing an operation. Typically, this is a single "carry-in" bit that is the stored carry-out from a previous ALU operation.

## 4. Implementation:

An ALU is usually implemented either as a stand-alone integrated circuit (IC), such as the 74181, or as part of a more complex IC. In the latter case, an ALU is typically instantiated by synthesizing it from a description written in VHDL, Verilog or some other hardware description language. For example, the following VHDL code describes a very simple 8-bit ALU:

```
entity alu is
port (  -- the alu connections to external circuitry:
  A  : in signed(7 downto 0);   -- operand A
  B  : in signed(7 downto 0);   -- operand B
  OP : in unsigned(2 downto 0); -- opcode
  Y  : out signed(7 downto 0));  -- operation result
end alu;


architecture behavioral of alu is
begin
 case OP is  -- decode the opcode and perform the operation:
  when "000" =>  Y <= A + B;   -- add
  when "001" =>  Y <= A - B;   -- subtract
  when "010" =>  Y <= A - 1;   -- decrement
  when "011" =>  Y <= A + 1;   -- increment
  when "100" =>  Y <= not A;   -- 1's complement
  when "101" =>  Y <= A and B; -- bitwise AND
  when "110" =>  Y <= A or B;  -- bitwise OR
  when "111" =>  Y <= A xor B; -- bitwise XOR
  when others => Y <= (others => 'X');
 end case;
end behavioral;
```

## 5. Future work

We have now completed a detailed study into how an ALU works. The next steps would be to take a deep dive into how operations are performed by the ALU along with all other components of the CPU, i.e. How an instruction cycle works.

**References:**


[1] Arithmetic logic unit - Wikipedia

[2] Computer Organization and Architecture 10th - William Stallings.pdf