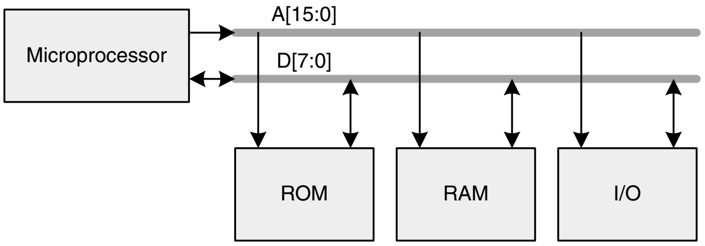

FIGURE 3.3 Microprocessor buses.

A microprocessor's entire address space is never occupied by a single function; rather, it is shared by ROM, RAM, and various I/Os. Each device is *mapped* into its own region of the address space and is enabled only when the microprocessor asserts an address within a device's mapped region. The process of recognizing that an address is within a desired region is called *decoding*. Address decoding logic is used to divide the overall address space into smaller sections in which memory and I/O devices can reside. This logic generates individual signals that enable the appropriate device based on the state of the address bus so that the devices themselves do not need any knowledge of the specific computer's unique address decoding.

3.2 MICROPROCESSOR INTERNALS

The multitude of complex tasks performed by computers can be broken down into sequences of simple operations that manipulate individual numbers and then make decisions based on those calculations. Certain types of basic instructions are common across nearly every microprocessor in existence and can be classified as follows for purposes of discussion:

- Arithmetic: add or subtract two values
- Logical: Boolean (e.g., AND, OR, XOR, NOT, etc.) manipulation of one or two values
- Transfer: retrieve a value from memory or store a value to memory
- Branch: jump ahead or back to a particular instruction if a specified condition is satisfied

Arithmetic and logical instructions enable the microprocessor to modify and manipulate specific pieces of data. Transfer instructions enable these data to be saved for later use and recalled when necessary from memory. Branch operations enable instructions to execute in different sequences, depending on the results of arithmetic and logical operations. For example, a microprocessor can compare two numbers and take one of two different actions if the numbers are equal or unequal.

Each unique instruction is represented as a binary value called an *opcode*. A microprocessor fetches and executes opcodes one at a time from program memory. Figure 3.4 shows a hypothetical microprocessor to serve as an example for discussing how a microprocessor actually advances through and executes the opcodes that form programs.

A microprocessor is a synchronous logic element that advances through opcodes on each clock cycle. Some opcodes may be simple enough to execute in a single clock cycle, and others may take multiple cycles to complete. Clock speed is often used as an indicator of a microprocessor's performance. It is a valid indicator but certainly not the only one, because each microprocessor requires a different number of cycles for each instruction, and each instruction represents a different quantity of useful work.

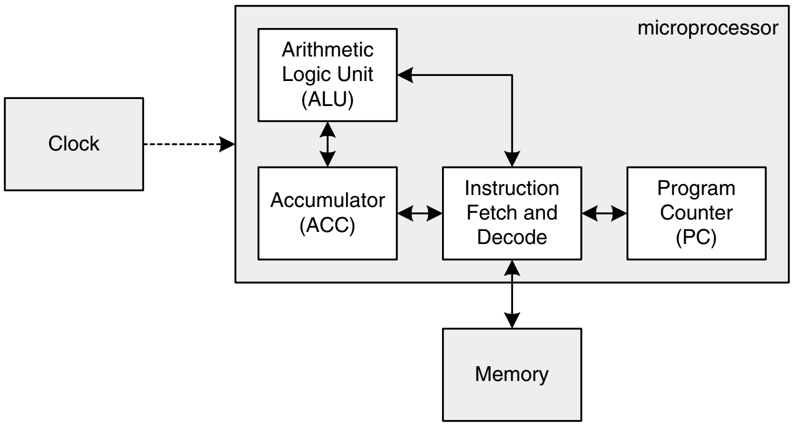

FIGURE 3.4 Simple microprocessor.

When an opcode is fetched from memory, it must be briefly examined to determine what needs to be done, after which the appropriate actions are carried out. This process is called *instruction decoding*. A central logic block coordinates the operation of the entire microprocessor by fetching instructions from memory, decoding them, and loading or storing any data as required. The *accumulator* is a register that temporarily holds data while it is being processed. Execution of an instruction to load the accumulator with a byte from memory would begin with a fetch of the opcode that represents this action. The instruction decoder would then recognize the opcode and initiate a memory read via the same microprocessor bus that was used to fetch the opcode. When the data returns from memory, it would be loaded into the accumulator. While there may be multiple distinct logical steps in decoding an instruction, the steps may occur simultaneously or sequentially, depending on the architecture of the microprocessor and its decoding logic.

The accumulator is sized to hold the largest data value that the microprocessor can handle in a single arithmetic or logical instruction. When engineers talk of an 8-bit or 32-bit microprocessor, they are usually referring to the internal *data-path* width—the size of the accumulator and the *arithmetic logic unit* (ALU). The ALU is sometimes the most complex single logic element in a microprocessor. It is responsible for performing arithmetic and logical operations as directed by the instruction decode logic. Not only does the ALU add or subtract data from the accumulator, it also keeps track of status flags that tell subsequent branch instructions whether the result was positive, negative, or zero, and whether an addition or subtraction operation created a carry or borrow bit. These status bits are also updated for logical operations such as AND or OR so that software can take different action if a logical comparison is true or false.

For ease of presentation, the microprocessor in Fig. 3.4 is shown having a single general-purpose accumulator register. Most real microprocessors contain more than one internal register that can be used for general manipulation operations. Some microprocessors have as few as one or two such registers, and some have dozens or more than a hundred. It is the concept of an accumulator that is discussed here, but there is no conceptual limitation on how many accumulators or registers a microprocessor can have.

A microprocessor needs a mechanism to keep track of its place in the instruction sequence. Like a bookmark that saves your place as you read through a book, the *program counter* (PC) maintains the address of the next instruction to be fetched from program memory. The PC is a counter that can be reloaded with a new value from the instruction decoder. Under normal operation, the microprocessor