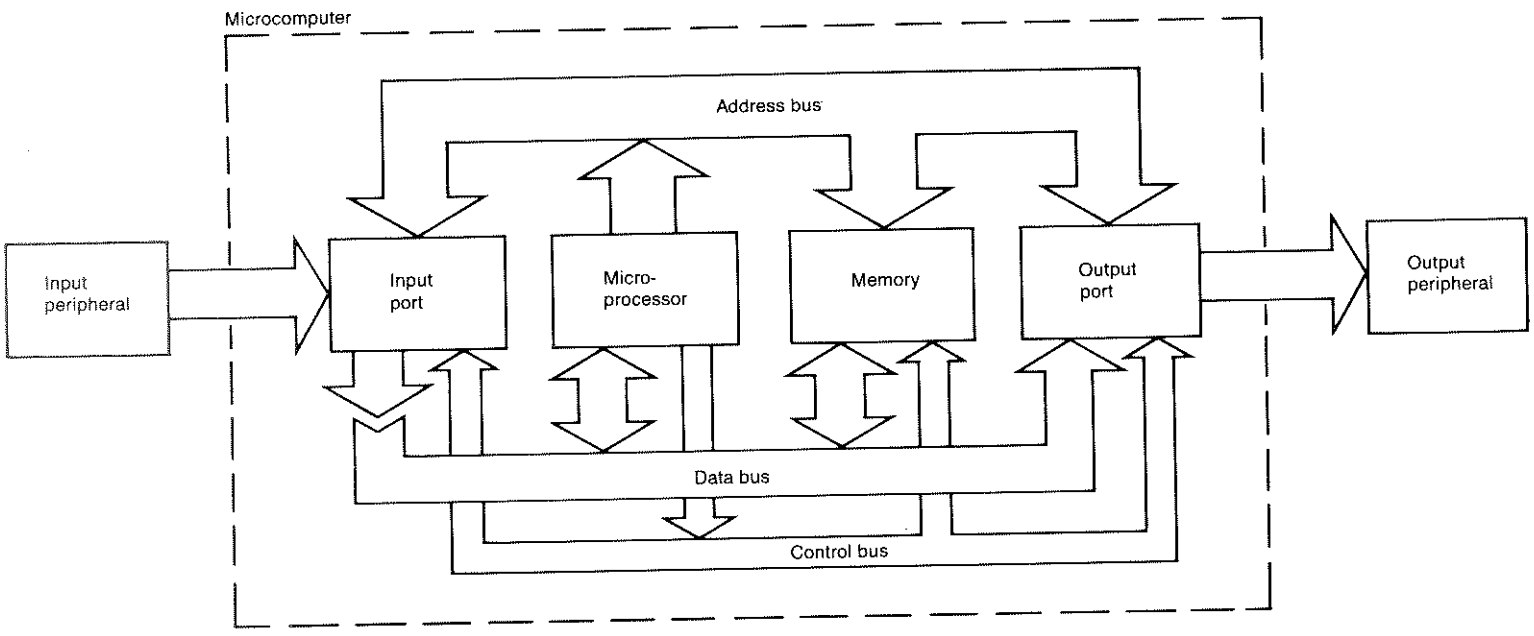

Figure 1.11. Basic microprocessor system block diagram.

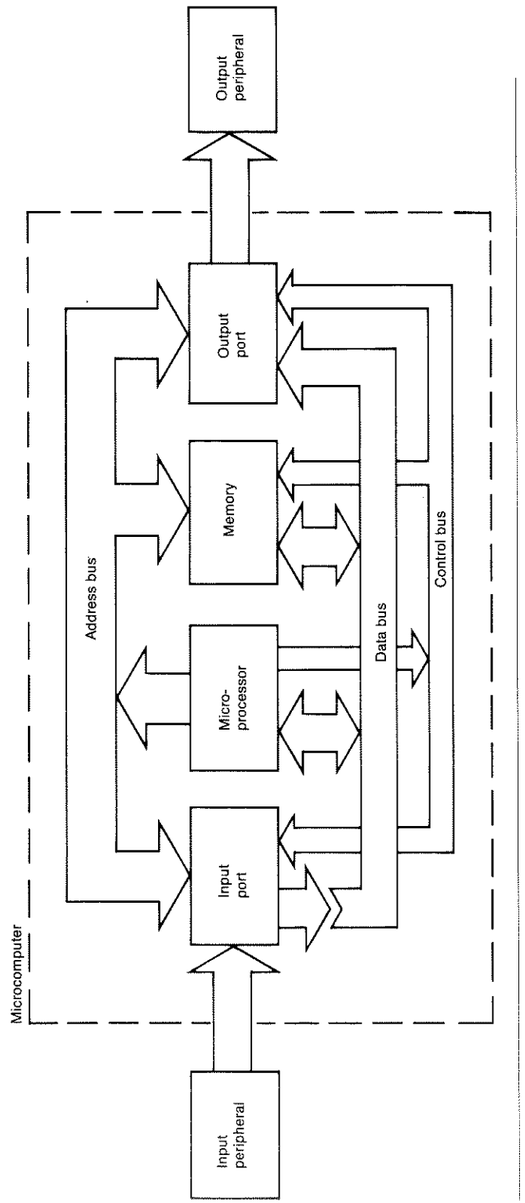

Figure 1.11. Basic microprocessor system block diagram.

exactly 64K bits. In this text, a lowercase k is used for the traditional "kilo" (1000) and an uppercase K for 1024.

1.2.6 Microprocessor System Architecture

All microprocessor systems include a microprocessor, memory, and I/O devices interconnected by address, data, and control buses, as shown in Fig. 1.11. A microprocessor cannot do anything by itself; it must be connected to memory and I/O devices. A complete, functional system is called a *microcomputer.* A *single-chip microcomputer* is a complete system on a single IC.

The basic operation of all microprocessor systems is the same, regardless of the type of microprocessor or the task being performed. The microprocessor reads an instruction from the memory, executes that instruction, and then reads the next instruction. This sequence repeats indefinitely as long as the system is running. Reading an instruction from memory is called the instruction *fetch,* and the sequence described above is called the *fetch-execute* sequence. (Some microprocessors use *prefetch*; they fetch a few instructions ahead of the instruction to be executed.)

Figure 1.12 illustrates the signals that connect to a typical microprocessor. This example shows an 8-bit microprocessor, which has eight wires in the data bus. The data bus signals are bidirectional, since data can flow into or out of the microprocessor. The address signals are outputs only, since the microprocessor generates the address for all operations. Most 8-bit microprocessors have 16-bit address buses, which allows them to address 64K (2^{16}) memory locations. Input and output ports are addressed similarly; subtle differences in I/O addressing techniques are described in Secs. 5.2.1 and 5.2.5.

The two fundamental control signals are $\overline{\text{READ}}$ and $\overline{\text{WRITE}}$, often called the *read strobe* and *write strobe*. The read strobe is asserted by the microprocessor to read data from a memory location or input port. The write strobe is asserted to write data to a memory location or output port. There are many variations of these signals, as described in Chap. 3, but the basic function is always the same. Additional control signals, including RESET, CLOCK, READY/$\overline{\text{WAIT}}$, BUS REQUEST, BUS ACKNOWLEDGE, and various types of interrupts are described briefly in Sec. 1.2.7, and in detail in Chap. 3.

Many types of memory devices are used in microprocessor systems. *Main memory* is directly addressed by the microprocessor and is required in all microprocessor systems. Chap. 4 describes types of main memory in detail. *Mass-storage* devices, such as disk drives, are not directly addressable but have large storage capacities at a relatively low cost. Mass-storage devices are covered in Chap. 9.

The two principal types of main memory are *read-only memory (ROM)* and *read/write memory (RAM).* ROMs are programmed with the desired data pattern before being installed in the application system. ROMs that are programmed by the IC manufacturer as part of the manufacturing process are called *mask-programmed.* The microprocessor cannot write data to a ROM; it is used only for permanent programs and unchanging data. *EPROMs* are erasable and programmable read-only memories that can be erased with ultraviolet light and programmed by the user with an instrument called an *EPROM programmer.*

With a read/write memory (RAM), the microprocessor can write data into the memory and read it back later. RAM is *volatile,* however: the contents are retained only as

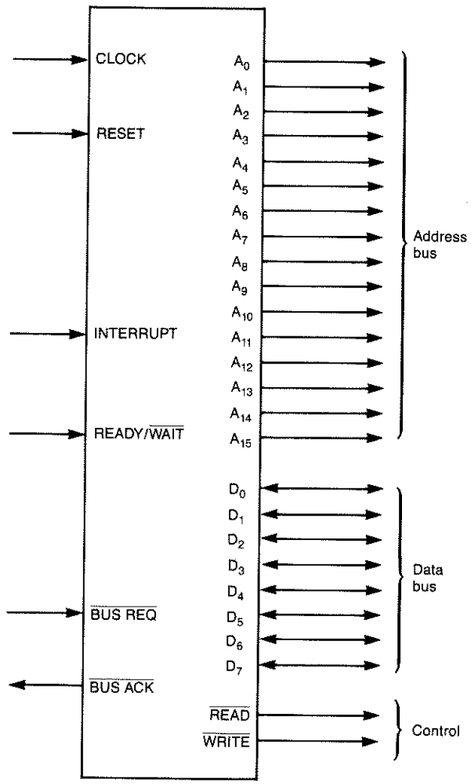

Figure 1.12. Basic microprocessor signals.

long as power is present. (Note that RAM seems like a strange acronym for read/write memory. RAM actually stands for random-access memory, but the term is misleading because ROMs are also random-access memories. However, RAM is widely used to designate read/write random access memory.)

The buses link the microprocessor to the rest of the system. The microprocessor selects a particular device via the address bus and then reads or writes data via the data bus. The data bus is a common, bidirectional bus used for all data transfers. Each *bus cycle* consists of a transfer of a word of data between the microprocessor and a memory or I/O device.

Each bus cycle begins when the microprocessor outputs an address to select one memory location or I/O port, as shown in Fig. 1.13. In this timing diagram, the address and data buses are each represented by a pair of lines. These indicate that the information on the bus is stable; when the lines on the timing diagram cross, it indicates that the data has changed. A single dashed line is used to show when the data bus is not being driven by any device and is therefore floating.

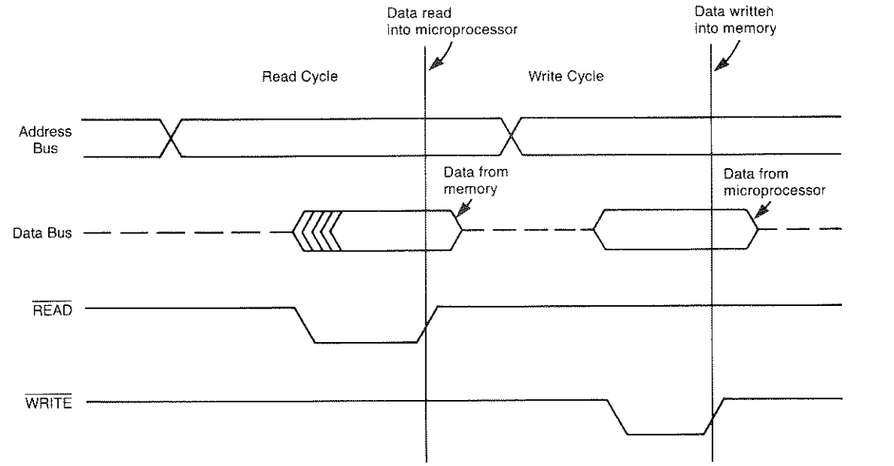

Figure 1.13. Basic bus timing.

The address is decoded by an *address decoder*, as shown in Fig. 1.14, which generates an address select signal for each device in the system. Every I/O port and memory location has a unique address, and the address decoder ensures that only one device is selected at a time.

Because all devices share the same data bus, when a device's address is not present on the address bus (along with the proper control signals), the device must be electrically removed from the data bus. This is accomplished with three-state drivers. Three-state drivers are included in all devices intended to directly drive the data bus, such as microprocessors, memories, and input ports. These drivers are enabled by the device's chip select input. This signal is sometimes called *OUTPUT ENABLE (OE)* if its only function is to control the output drivers. When a device's output drivers are enabled, it is said to be *driving* the bus.

If two devices attempt to drive the data bus at the same time, a *bus conflict* occurs and neither device is likely to control the bus successfully. If a bus conflict occurs while the data on the bus must be valid, invalid data can be read from or written to memory. Even a bus conflict that occurs when data does not need to be valid can cause problems. If one driver is trying to pull a line low and another is trying to pull it high, large currents can flow through both devices. This can cause a drop in the system's power supply voltage, which can, in turn, cause memory errors and many other mysterious failures.

After the microprocessor outputs the address, it asserts either $\overline{\text{READ}}$ or $\overline{\text{WRITE}}$ to

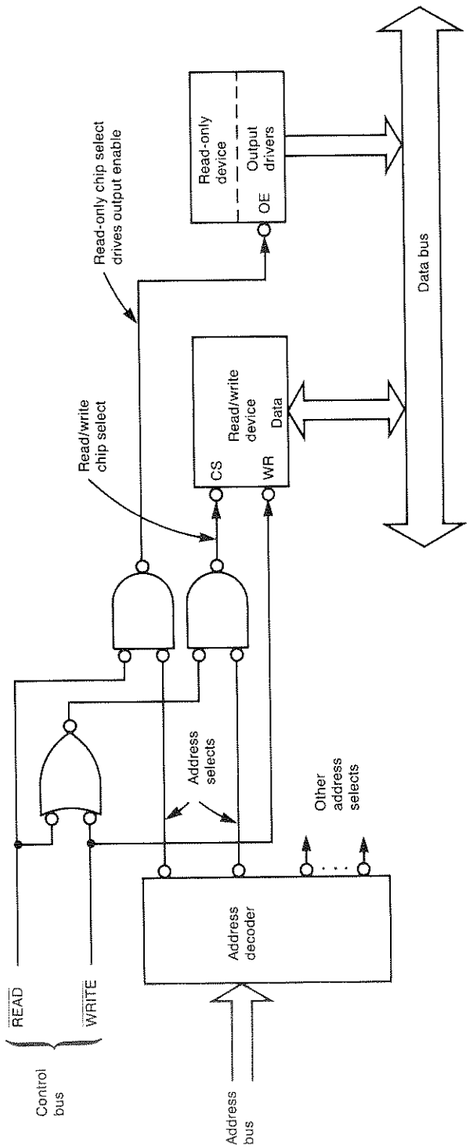

Figure 1.14. Generation of address select and chip select signals.

indicate when the addressed device should drive data on (or read data from) the data bus. For a read-only device, $\overline{\text{READ}}$ is ANDed with $\overline{\text{ADDRESS SELECT}}$ to generate the $\overline{\text{CHIP SELECT}}$ ($\overline{\text{CS}}$) signal. $\overline{\text{CS}}$ enables the device's output drivers, which thus are enabled only when the device's address is present on the address bus and the read strobe is asserted.

For read/write devices, $\overline{\text{ADDRESS SELECT}}$ is ANDed with $\overline{\text{READ}}$ OR $\overline{\text{WRITE}}$. Thus, $\overline{\text{CS}}$ is asserted whenever either operation occurs. $\overline{\text{WRITE}}$ also connects directly to the I/O or memory device. If $\overline{\text{CS}}$ is asserted and $\overline{\text{WRITE}}$ is false, the device's output drivers are enabled. However, if $\overline{\text{WRITE}}$ is also asserted, then the device reads data from the bus instead of outputting data to it.

The actual data transfer occurs as $\overline{\text{READ}}$ or $\overline{\text{WRITE}}$ is negated to allow as much time as possible for the addressed device to decode the address and prepare to perform the read or write operation. The addressed device needs time to decode the address, select the desired memory location, and drive the data onto the bus (for a read cycle) or accept data from the bus (for a write cycle).

1.2.7 System Control Signals

Most microprocessors have a number of control signals in addition to the basic read and write strobes. This section provides brief descriptions of the basic functions of these signals; more detailed descriptions are included in Chap. 3.

The CLOCK input to the microprocessor paces the operation of the system. This signal is typically generated by a crystal oscillator. Each instruction execution requires a fixed number of clock cycles as determined by the design of the microprocessor.

The RESET input starts (or restarts) the microprocessor. It causes the microprocessor to begin executing instructions from a predefined memory location. With some microprocessors, execution begins from address zero when RESET is asserted; others use a different reset address, but in some way it is always well defined.

Interrupts are input signals that force a change in the software execution sequence. When an INTERRUPT input is asserted, the software is forced to jump from whatever program it is executing to a special program called the *interrupt service routine*. Interrupts are useful because they allow the system to provide fast response to real-time events. Many microprocessors have two or more interrupt inputs and an interrupt acknowledge output.

The $\overline{\text{WAIT}}$ input allows memory and I/O devices to force the microprocessor to lengthen the bus cycle. This is necessary when the memory or I/O device is too slow to respond to the microprocessor in the time it normally allows. When $\overline{\text{WAIT}}$ is asserted, the microprocessor freezes the state of the buses for one clock cycle; this is called a *wait state*. If the $\overline{\text{WAIT}}$ input is kept asserted, additional wait states are generated. When $\overline{\text{WAIT}}$ is negated, the microprocessor continues operation. This signal is called READY on some microprocessors.

The $\overline{\text{BUS REQ}}$ (bus request) input forces the microprocessor to relinquish control of the address, data, and control buses so that another microprocessor or other device (such as a direct memory access controller, as described in Sec. 5.7) can take control. The microprocessor asserts the $\overline{\text{BUS ACK}}$ (bus acknowledge) output to indicate that it has released the buses and the requesting device can take over.

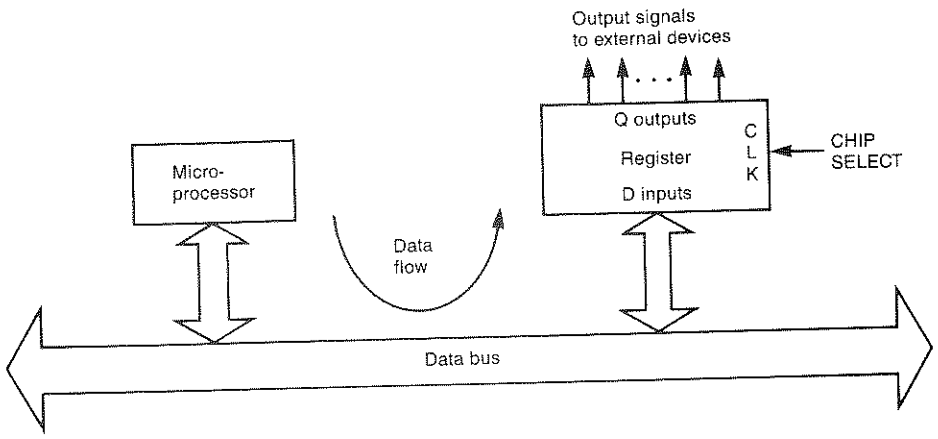

Figure 1.15. Basic output port.

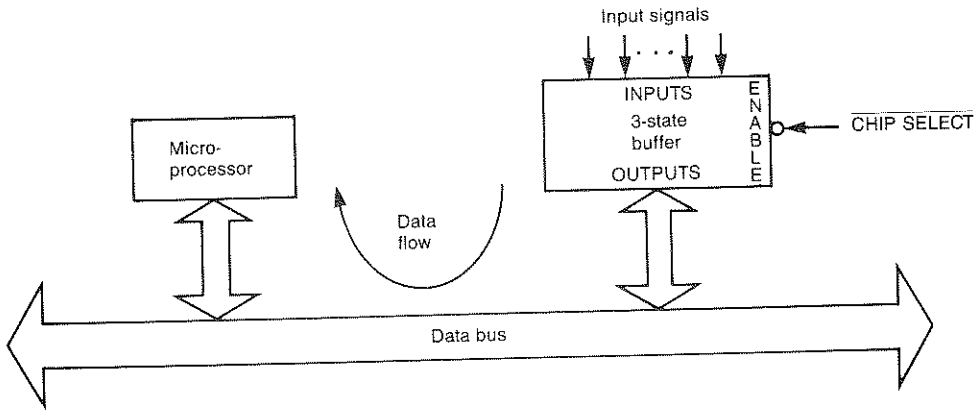

Figure 1.16. Basic input port.

1.2.8 Basic I/O Ports

Input and output ports interface the microprocessor to devices that cannot connect directly to the buses. An output port is fundamentally just a register, as shown in Fig. 1.15. When the microprocessor writes to the address assigned to the port, the port stores the data from the data bus. Thus, it provides latched outputs to external devices, which are changed whenever the microprocessor writes to the port.

An input port is fundamentally just a three-state driver, as shown in Fig. 1.16. When the microprocessor reads from the address assigned to the port, the three-state driver drives the data from the external inputs onto the data bus. The microprocessor then reads the signals from the bus.

The output-port register and input-port three-state driver may be implemented with standard TTL or CMOS devices. They may also be incorporated with additional logic

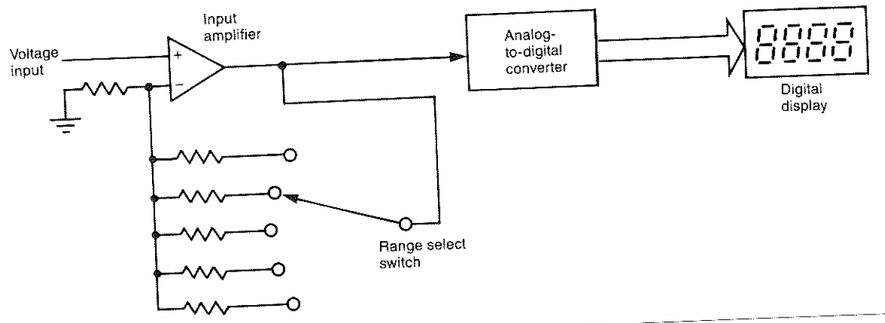

Figure 1.18. Basic digital voltmeter block diagram.

The addition of a microprocessor allows new features to be added and results in a fundamental change in the internal design of the instrument. Figure 1.19 shows the block diagram for a microprocessor-based digital voltmeter. The A/D converter connects not to the display but to the microprocessor's buses. The microprocessor executes a program that causes it to read the data from the A/D converter and output it to the display. Thus, the data flow is determined by the software, which must instruct the hardware to read from the A/D converter and then write to the display.

The rotary switch for setting the range is replaced with momentary push buttons. To select a voltage range the user presses a button, which is sensed by the microprocessor via an input port. The microprocessor then writes data to an output port, which in turn controls the gain of the input amplifier. The push-button switches are less expensive and more reliable than the rotary switch, but this approach has more important advantages as well. The switches do not directly control the gain but provide inputs to the microprocessor system, which in turn sets the gain. This allows the software to decide what to set the gain to, based on the setting of the switches and any other data.

Now that the microprocessor has control of the gain, automatic range selection *(autoranging)* can be implemented with only some additional software. The microprocessor begins by selecting minimum gain. It then reads the A/D converter output, and if the value read is small, the gain is increased. Thus, via a procedure implemented in software (called an *algorithm*), the microprocessor can automatically select the optimum gain setting. The software can also perform additional calculations, allowing such functions as display of the reading as a percentage of a previously stored value.

Remote control is another feature made practical by microprocessor-based design. A microprocessor-based voltmeter can connect to a computer that can read the voltage and also send commands to the meter to select the range and other functions. During remote control operation, the voltmeter's control software takes its commands from (and sends data to) the remote control interface rather than the front panel.

This example illustrates the advantages typically gained from microprocessor-based design: improved user interface, more automatic measurements, remote control capability, and the ability to perform calculations on raw measurement data before it is displayed.

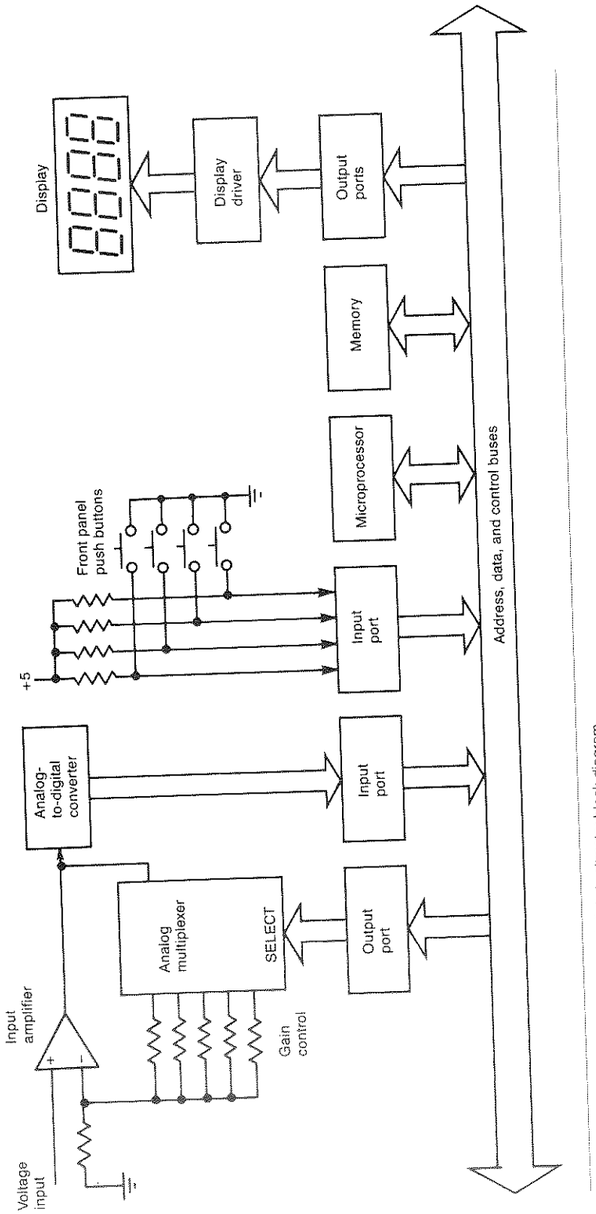

Figure 1.19. Microprocessor-based digital voltmeter block diagram.