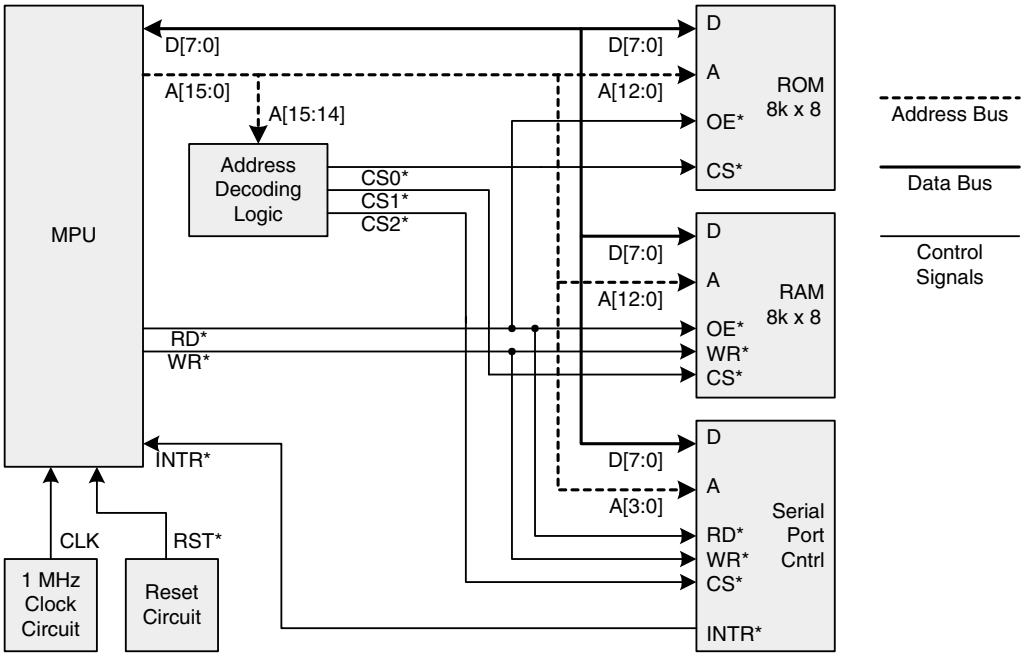

FIGURE 3.7 Eight-bit computer block diagram.

hypothetical computer are active-low, as are the control signals in most computer designs that, according to convention, have been in widespread use for the past few decades. Active-low signal names have some type of symbol as a prefix or suffix to the signal name that distinguishes them from active-high signals. Common symbols used for this purpose include #, *, –, and _. From a logical perspective, it is perfectly valid to use active-high signaling. However, because most memory and peripheral devices conform to the active-low convention, it is often easier to go along with the established convention.

While hypothetical, the microprocessor shown contains characteristics that are common in off-the-shelf eight-bit microprocessors. It contains an 8-bit data bus and a 16-bit address bus with a total address space of 64 kB. The combined MPU bus, consisting of address, data, and control signals, is asynchronous and is enabled by the assertion of read and write enable signals. When the microprocessor wants to read a location in memory, it asserts the appropriate address along with RD* and then takes the resulting value driven onto the data bus. As shown in the diagram, memory chips usually have *output enable* (OE*) signals that can be connected to a read enable. Such devices continuously decode the address bus and will emit data whenever OE* is active.

Not all 64 kB of address space is used in this computer. Address decoding logic breaks the single 64-kB space into four 16-kB regions. According to the state of A[15:14], one and only one of the chip select signals is activated. The address decoding follows the truth table shown in Table 3.1 and establishes four address ranges.

Once decoded into regions, A[13:0] provides unique address information to the memory and I/O devices connected to the MPU bus. One memory region, the upper 16 kB, is currently left unused. It may be used in the future if more memory or another I/O device is added. Each memory and I/O de-

TABLE 3.1 Address Decoding Truth Table

A[15]	A[14]	Chip Select	Address Range
0	0	CS0*	0x0000-0x3FFF
0	1	CS1*	0x4000-0x7FFF
1	0	CS2*	0x8000-0xBFFF
1	1	none	0xC000-0xFFFF

vice has a chip select input and will respond to a read or write command only when that select signal is active. Furthermore, each chip, including the microprocessor, contains internal tri-state buffers to prevent contention on the bus. The tri-state buffers are not enabled unless the chip's select signal is active and a read is being performed (a write, in the case of the microprocessor). Without external address decoding, none of these chips can share an address region with any other devices, because they do not have enough address bits to fully decode the entire 16-bit address bus.

Not all address bits are used by the memory and serial port chips. The ROM and RAM are each only 8k in size. Therefore, only 13 address bits, A[12:0], are required and, as a result, A[13] is left unconnected. The serial port has far fewer memory locations and therefore uses only A[3:0], for a maximum of 16 unique addresses.

When a device does not utilize all of the address bits that have been allocated for its particular address region, the potential for *aliasing* exists. The ROM occupies only 8k (13 bits) of the 16k (14 bits) address region. Therefore, the ROM has no knowledge of any additional addresses above 8k: the region from 0x2000 to 0x3FFFF. What happens if the MPU tries to read location 0x2000? 0x2000 differs from 0x0000 only in the state of A[13]. Because the ROM does not have any knowledge of A[13], it interprets 0x2000 to be 0x0000. In other words, 0x2000 *aliases* to 0x0000. Similarly, the entire upper 8k of the address region aliases to the lower 8k. In the case of the serial port controller, there is a greater degree of aliasing, because the serial port only uses A[3:0]. This means that there can be only 16 unique address locations in the entire 16k region. These 16 locations will therefore appear to be replicated 2^{10} = 1,024 times as indicated by the ten unused address bits, A[13:4].

As long as the software is properly written to understand the computer's memory map, it will properly access the memory locations that are available and will avoid aliased portions of the memory map. Aliasing is not a problem in itself but can lead to problems if software does not access memory and peripherals in the way in which the hardware engineer intended. If software is written for the hypothetical computer with the incorrect assumption that 16 kB of RAM is present, data may be unwittingly corrupted when addresses between 0x6000 and 0x7FFF are written, because they will alias to 0x4000-0x5FFF and overwrite any existing data.

When the MPU wants to read data from a particular memory location, it asserts that address onto A[15:0]. This causes the address decoder to update its chip select outputs, which enables the appropriate memory chip or the serial port. After allowing time for the chip select to propagate, the RD* signal is asserted, and the WR* signal is left unasserted. This informs the selected device that a read is requested. The device is then able to drive the data bus, *D[7:0],* with the requested data. After allowing some time for the read data to be driven, the MPU captures the data and releases the RD* signal, ending the read request. The sequence of events, or timing, for the read transaction is shown in Fig. 3.8.

This type of MPU bus is asynchronous, because its sequence of events is not driven by a clock but rather by the assertion and removal of the various signals that are timed relative to one another by the