



Figure 11.2 Illustration of a virtual memory system that divides an address space among two physical memories. The MMU uses an address to decide which memory to access.

11.6 Address Translation Or Address Mapping

Each of the underlying memory systems in Figure 11.2 operates like the physical memory systems discussed in Chapter 10: the hardware expects requests to reference addresses beginning at zero. Thus, each of the two memories recognizes the same set of addresses. For memory 1, the virtual addresses associated with the memory cover the same range as the hardware expects, which means the MMU can pass a request from the processor to memory 1 with no changes. For memory 2, however, the processor generates addresses starting at 1000, so the MMU must *map* an address to the lower range (i.e., real addresses 0 through 999) before passing a request to memory 2. We say that the MMU *translates* the address.

Mathematically, address mapping for memory 2 is straightforward: the MMU merely subtracts 1000 from an address. Figure 11.3 explains the concept.

```

receive memory request from processor;
let A be the address in the request;
if (A >= 1000) {
    A = A - 1000;
    pass the modified request to memory 2;
} else {
    pass the unmodified request to memory 1;
}

```

Figure 11.3 The sequence of steps used by a Memory Management Unit to create the virtual memory depicted in Figure 11.2. The MMU maps the virtual address space onto two physical memories.