0.1 Physical and Virtual Addressing

The main memory of a computer is organized as an array of M contiguous byte-sized cells. Each byte has a *physical address* (PA). The first byte has an address of 0, the next byte has an address of 1, etc. Given this simple organization, it makes sense for a CPU to access memory using physical addresses, an approach known as *physical addressing*.

Figure 0.1 demonstrates an example in the context of a load instruction that reads the 4-byte word starting at physical address 4. When the CPU executes the load instruction, it generates an effective physical address and passes it to the main memory over the memory bus. The main memory fetches the 4-byte word starting at physical address 4 and returns it to the CPU, which stores it in a register.

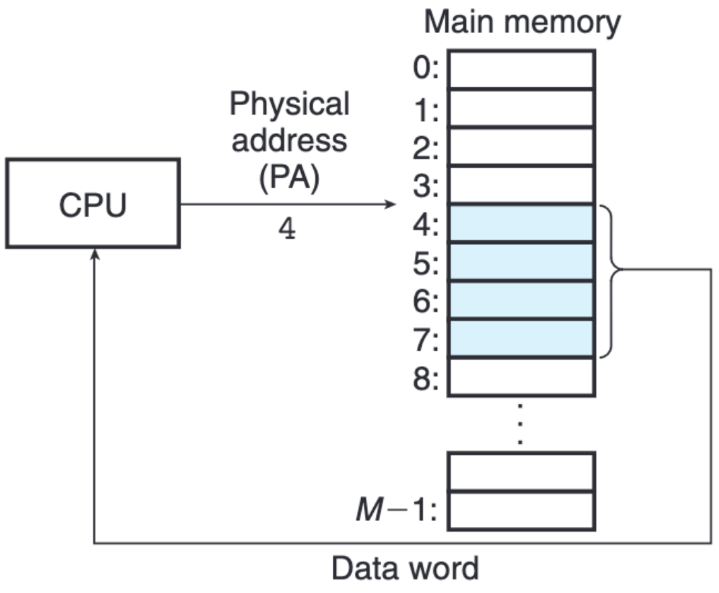

Figure 0.1: A system that uses physical addressing

In contrast, most modern processors use a form of addressing known as *virtual addressing*, shown in Figure 0.2. With virtual addressing, the CPU accesses main memory by generating a *virtual address* (VA), which is converted to the appropriate physical address before being sent to main memory. The task of converting a virtual address to a physical one is known as *address translation*.

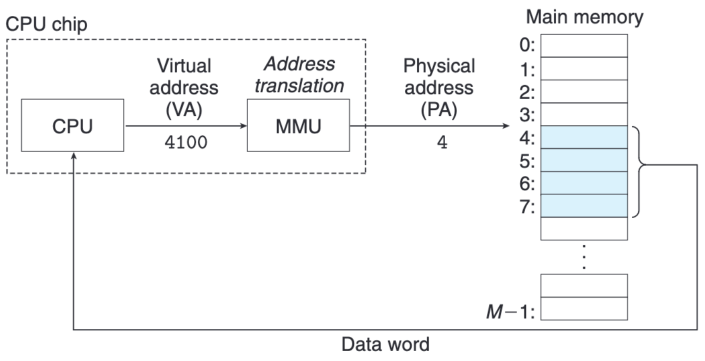

Figure 0.2: A system that uses virtual addressing

Like exception handling, address translation requires close cooperation between the CPU and the operating system. Dedicated hardware on the CPU chip called the *memory management unit* (MMU) translates virtual addresses on the fly, using a lookup table stored in main memory whose contents are managed by the operating system.