Now a memory "get" from RAM has three stages:

```
return addressSpace.get(address);
    return blocks[i >>> 12].get(address & 0xfff);
        return memory[address];
```

And one from a higher address has four:

```
return addressSpace.get(address);
    return blocks[i >>> 22][(i >>> 12) & 0x3ff].get(address & 0xfff);
        return memory[address];
```

This two-layer optimization has saved us memory while avoiding the production of a bottleneck in every RAM memory access. Each call and layer of indirection in a memory "get" performs a function. This is indirection the way it should be used—not for the sake of interfacing, but to achieve the finest balance of performance and footprint.

Lazy initialization is also used in JPC wherever there is chance of storage never being used. Thus a new JPC instance has a physical address space with mappings to `Memory` objects that occupy no space. When a 4 KB section of RAM is read from or written to for the first time, the object is fully initialized, as in Example 9-1.

EXAMPLE 9-1. Lazy initialization

```
public byte getByte(int offset)
{
    try {
        return buffer[offset];
    } catch (NullPointerException e) {
        buffer = new byte[size];
        return buffer[offset];
    }
}
```

The Perils of Protected Mode

The arrival of protected mode brings a whole new system of memory management into play, with another layer of complexity added on top of the physical address space. In protected mode, paging can be enabled, which allows the rearrangement of the constituent 4 KB blocks of the physical address space. This rearrangement is controlled by a sequence of tables held in memory that can be dynamically modified by the code running on the machine. Figure 9-4 shows the path followed on a complete page translation.

In principle, every memory access on the machine will require a full lookup sequence through this paging structure to find the physical address that the given linear address maps to. As this process is so convoluted and costly, a real machine will cache the result of these lookups in translation look-aside buffers (TLBs). In addition to these added layers of indirection, there are extra protection features with memory paging. Each mapped page can be given a user or supervisor and a read or read/write status. Code that attempts to access pages without sufficient