```
                *instruction = PENPOWER_SIMPLIFIED;
            }
            else {
                *instruction = PENPOWER_TRADITIONAL;
            }
        }
    }
```

5. 同步 Cache 与主存

至此，从编程语言的视角来看，已经完成了修改指令的全部工作，但是事实上还有一个更底层的细节需要关注，就是 Cache 与主存的同步。

由于 Cache 的存在，修改很可能没有立即被反应到主存，尤其是在 ARM9 这类哈佛体系结构的处理器中，指令和数据使用不同的 Cache，情况就更加复杂，因此必须确保所修改的指令在执行前已经被正确同步到主存，而不是仅仅存放在数据 Cache 中。

对于 ARM 来说，可以使用协处理器指令来同步主存与 Cache，但这意味着需要在高级语言中嵌入汇编代码，而由此引起的另一个更大的麻烦是，不同的 ARM 核的 Cache 清理指令略有不同，这使得汇编级代码不利于移植。

不过幸运的是，我们发现了一个 ***mprotect()*** 的有益的副作用，即清理 Cache，使其将修改后的脏数据回写主存

调用 ***mprotect()*** 时，如果指定内存页将要设置的权限与当前权限不同，则会触发 Cache 与主存的同步。与使用协处理器指令相比，这种方法不需要嵌入汇编，因而也不需要为不同的 ARM 核编写不同的汇编代码。

6. 结束语

本例中，通过运行时修改指令，实现了在现有二进制库的基础上查询简体或者繁体中文词语的功能。事实上，在其他平台上，动态指令修改早已被广泛地使用于例如防火墙，病毒，蠕虫，加密等诸多应用中。不过，仍然需要强调的是，动态修改指令会产生了潜在风险，如果在不正确的地址产生了不正确的指令，则会导致无效指令，访问违例等崩溃，因此需要特别谨慎。