FlushInstructionCache Flush the instruction cache

#include <Memory.h> **Memory Manager**

void **FlushInstructionCache( );**

The **FlushInstructionCache** function flushes the current contents of the instruction cache. Because flushing this cache degrades performance of the CPU, this routine should be called only when absolutely necessary.

Notes: Be sure to check that the trap _HWPriv is implemented before calling this routine. See **Using the Gestalt Manager** for sample code that shows how to determine whether a trap is implemented. If you call this routine and _HWPriv is not implemented, you application will crash.

The MC68020, MC68030 and MC68040 microprocessors have instruction caches.

Any time you modify part of executable code, you risk creating stale instructions in the instruction cache. A microprocessor that contains an instruction cache store the most recently executed instructions in its internal instruction cache, separate from main memory. Whenever your code modifies itself or any data in memory that contains executable code, there is a possibility that a copy of the modified instructions will be in the instruction cache. If so, attempting to execute the modified instructions actually results in the execution of the cached instructions, which are stale.

Stale instructions can be avoided by flushing the instruction cache every time you modify executable instructions in memory. Flushing the cache invalidates all entries in it and forces the processor to refill the cache from main memory.

If you want to flush the instruction cache of a processor with a copy-back data cache, such as the MC68040 processor, you may want to write your own routine that calls the _CacheFlush trap in assembly language. This trap flushes the data cache before it flushes the instruction cache to insure that any instructions subsequently copied to the instructions cache are not copied from stale RAM.

```
// FlushCache
// Flush the CPU cache(s). This is required on the 68040 after modifying
// code.

#define _CacheFlushTrap 0xA0BD

void FlushCache(void)
{
    if (TrapAvailable( _CacheFlushTrap))
        asm
        {
            dc.w _CacheFlushTrap
        }
}
```

The preceding code example calls a routine called **TrapAvailable**. See **Using the Gestalt Manager** for the definition of this routine.