The SURE Architecture - big memory

David May: 4 November, 2019

Caches

The original description of the SURE Architecture (December 2016) assumed a simple memory system with no caches. This is appropriate for a real-time micro-controller, but raises a question: can we use similar ideas for large, cached, memory systems?

It seems that we can use a conventional set-associative cache, addressed using a pointer, not a physical memory address. The blocks in the cache hold blocks within objects; this is implemented either by aligning all objects in memory on block boundaries, or by re-aligning blocks when they are transferred between memory and cache. Assuming a cache block of four 32-bit words, when a pointer is used to access memory, bits 0:3 of pointer identify a byte address within a cache block; the exclusive-or of bits 4:15 and bits 16:31 identify the set associated with the access. This means that large objects will be distributed across many sets, and that a large number of small objects will also be distributed across many sets.

The directory and the cache can be accessed at the same time (in parallel). If there is a cache hit, the access will be completed within a single cycle; otherwise the directory information will be used to transfer a cache block from memory.

The cache blocks have a flag to indicate if the associated object has been marked. During the marking phase, if there is a cache hit on a load and the flag is clear, the flag will be set and the cache will cause the garbage collector to mark the associated object in the directory. During sweep, the garbage collector invalidates cache blocks associated with unmarked objects, and clears the mark flags. The garbage collector marking phase starts by causing the processor registers to be written back to the associated object(s) in memory; this means that the garbage collector and directory are actually part of the memory, not the processor.

An immediate consequence of this is that accesses that hit the cache will have the same performance as in a conventional memory system - they do not need to use the directory to calculate the address or to check that the address is valid. And if there is a cache miss, the directory information will be available to proceed with the memory access by the time that the cache miss is detected.

Shared memory

The garbage collector, directory and memory form an ‘Object memory’. This can be shared between several processors, each with its own cache system. This will involve shared access to the directory, with potential access delays (a few cycles at most) in the rare event of cache misses from several processors coinciding.

Large memory systems

A large memory system can be constructed from several smaller garbage collected memory systems. It is necessary to synchronise the operation of the garbage collectors at the beginning of the marking phase and at the beginning of the sweeping phase. Note that the garbage collectors do not need to synchronise with the application, so this will not introduce any pauses during execution of the application.

During the marking phase, there will be pointers stored in each memory that point to objects in other memories. When one of these is encountered, a mark message is sent to the memory containing the object that is pointed to, causing it to be marked.

Each memory has a completion signal and the overall marking phase is completed when all of the completion signals are active. Whenever a memory sends a mark message, it removes its completion signal until it receives a corresponding mark reply signal from the memory containing the marked object. Also, if a memory that is asserting its completion signal receives a mark message, it removes its completion signal and continues marking, prior to sending the mark reply message.

Objects can be freely distributed across the memories. It is probably a good idea to allocate a new object within the memory that has the most space; for a few memories this can be done so as to support simultaneous allocation requests from several processors.