Instruction	Description
I/O And Context Swap Instructions	
DRAM (read and write)	Move data between DRAM and ME
DRAM (RBUF and TBUF)	Move data between DRAM and RBUF/TBUF
CAP (CSR addressing)	Move data between CAP CSR and ME
CAP (calculated addressing)	Move data between CAP devices and ME
CAP (reflect)	Move data between registers in two MEs
CTX_ARB	Perform context swap and wake on event
HALT	Put ME to sleep and interrupt Xscale
HASH	Issue a request to the Hash Unit
MSF	Issue a request to the Media Switch Fabric
PCI	Issue a request to the PCI bus
SCRATCH (read and write)	Move data between MEs and Scratch Memory
SCRATCH (atomic operation)	Perform an atomic operation on Scratch Memory
SCRATCH (ring operation)	Insert or extract data from Scratch Ring
SRAM (read and write)	Move data between ME and SRAM
SRAM (atomic operation)	Perform an atomic operation on SRAM
SRAM (CSR)	Read or write an SRAM CSR
SRAM (read queue descriptor)	Access queue in SRAM
SRAM (write queue descriptor)	Change queue in SRAM
SRAM (enqueue)	Enqueue item in SRAM queue
SRAM (dequeue)	Dequeue item from SRAM queue
SRAM (ring operation)	Manipulate a communication ring in SRAM
SRAM (journal operation)	Perform journal operation in SRAM

Figure 19.2 Part 2 of the microengine (MEv2) instruction set.

19.6 Separate Memory Address Spaces

The microengine architecture differs from the XScale in another way: microengine hardware does not map memory or I/O devices into a linear address space. Thus, unlike the XScale, a microengine does not view memory as a seamless, uniform repository. Instead, a program running on a microengine must specify the exact memory for which a transfer is required.

Figures 19.1 and 19.2 illustrate how the concept of separate address spaces affects the architecture: the instruction set must include a separate instruction for each type of memory and each type of I/O device. For example, a *DRAM* instruction can access a