2.14 Simultaneous Multi-Threading

In order to improve instruction throughput, the processor implements Simultaneous Multi-Threading (SMT). Single-threaded applications do not always occupy all resources of the processor at all times. The processor can take advantage of the unused resources to execute a second thread concurrently.

Resources such as queue entries, caches, pipelines, and execution units can be competitively shared, watermarked, or statically partitioned in two-threaded mode (see Table 3 below).

These categories are defined as:

- Competitively Shared: Resource entries are assigned on demand. A thread may use all resource entries.
- Watermarked: Resource entries are assigned on demand. When in two-threaded mode a thread may not use more resource entries than are specified by a watermark threshold.
- Statically Partitioned: Resource entries are partitioned when entering two-threaded mode. A thread may not use more resource entries than are available in its partition.

Note that "Competitively Shared" is listed as the default protocol for the L3 cache, but sharing policy can be configured. See document 56375 "AMD64 Platform Quality of Service Extensions" and Processor Programming Reference for details.

Table 3. Resource Sharing

Resource	Competitively Shared	Watermarked	Statically Partitioned
L1 Instruction Cache	X		
ITLB	X		
Op Cache	X		
Dispatch Interface	X		
L1 Data Cache	X		
DTLB	X		
L2 Cache	X		
L3 Cache	X		
Integer Scheduler		X	
Integer Register File		X	
Load Queue		X	
Floating Point Physical Register	X		
Floating Point Scheduler		X	
Memory Request Buffers		X	
Op Queue			X
Store Queue			X
Write Combining Buffer		X	
Retire Queue			X

For partitioned resources, arbitration between threads is generally round-robin unless a given thread is stalled.

It is expensive to transition between single-threaded (1T) mode and dual-threaded (2T) mode and vice versa, so software should restrict the number of transitions. If running in 2T mode, and one thread finishes execution, it may be beneficial to avoid transitioning to 1T mode if the second thread is also about to finish execution.

If the two threads are running different code, they should run in different linear pages to reduce BTB collisions.

Two threads which concurrently run the same code should run at the same linear and physical addresses. Operating system features which randomize the address layout such as Windows® ASLR should be configured appropriately. This is to facilitate BTB sharing between threads.

2.15 LOCKs

The processor implements logic to improve the performance of LOCKed instructions. In order to benefit from this logic, the following guidelines are recommended:

- Ensure that LOCKed memory accesses do not cross 16-byte aligned boundaries.
- Following a LOCKed instruction, refrain from using floating point instructions as long as possible.
- Ensure that Last Branch Record is disabled (DBG_CTL_MSR.LBR)