

Deep Tech

THE FUTURE BEGINS TODAY

BY RIK MYSLEWSKI



MULTICORE MANAGEMENT

Last month, this space was filled with news about Nehalem, Intel's next generation of super-tiny, power-miserly, über-muscular, multicored processors. (If you missed my musings, you can find them on the Intertubes at www.maclife.com/article/tick_tock). Among the many advances of this data-wrestling wonder will be its ability to process two simultaneous streams of data and instructions—also known as threads—through each of its cores. In its first incarnation, Nehalem will have up to eight cores, so we're talking 16 threads coursing through this baby in a process called Simultaneous Multi-Threading (SMT).

That's a lot of data to manage efficiently and safely—"efficiently" meaning keeping each thread fed with the most important data at exactly the right nanosecond, and "safely" meaning keeping one thread from munging data that another thread is working with. Luckily, Intel smarties are all over this problem, specifically with a pair of interlocking technologies called Speculative Parallel Threading (SPT) and Transactional Memory (you guessed it: TM). [SPT is aimed squarely at the efficiency challenge, while TM tackles safety. Both work together to squeeze as much power as possible out of multicore processors.](#)

SPT takes calculated risks. After a human being writes software in a language such as C++, a software compiler turns that human-readable code into the stream of ones and zeros that a processor understands. The problem is that the compiler needs to be conservative about how it handles threads, since it doesn't know the state of the computer on which the compiled software will be running—the exact contents of its memory, for example. SPT takes a different approach, making guesses as to what parts of an application can run simultaneously on multiple threads, during the actual running of the software. When it's right, much time is saved because chunks of code are running simultaneously. When it's wrong, it simply dumps the bad results and tries again. If it's right more than it's wrong—and this is usually the case with SPT—more work gets done in less time, which is A Good Thing.

TM works hand in hand with SPT by making sure that one of those multiple threads isn't changing data that another thread has already "checked out" of memory. In essence, it does this by making each complete memory operation—read, modify, write—a separate transaction, so that another thread can't, say, modify the same bit of data before the first thread is done with that bit of data.

If this all just sounds like a *mélange* of mojo and juju to you, don't sweat it. Both SPT and TM will work behind the scenes in future processors, making the most of those multiple threads in multiple cores without you having to lift a finger to help. And how many threads are we talking about in future computers? According to one Intel researcher, "Hundreds, or in some cases thousands."

Rik remembers when "threads" referred to his five-button jacket and chinos with waist pleats and turn-ups.