The only real solution is to abandon the IA-32 as the main line of development and go to a completely new ISA. This is, in fact, what Intel intends to do. In fact, it has plans for two new lines. The EMT-64 is a wider version of the Pentium 4, with 64-bit registers and a 64-bit address space. This processor solves the address space problem but still has all the implementation complexities of the Pentium 4. It can best be thought of as a wider Pentium.

The other new architecture, developed jointly by Intel and Hewlett Packard, is called the **IA-64**. It is a full 64-bit machine from beginning to end, not an extension of an existing 32-bit machine. Furthermore, it is a radical departure from the Pentium 4 in many ways. The initial market is for high-end servers, but it may catch on in the desktop world eventually. In any case, the architecture is so radically different from anything we have studied so far that it is worth examining it just for that reason. The first implementation of the IA-64 architecture is the Itanium series. In the remainder of this section we will study the IA-64 architecture and the Itanium 2 CPU that implements it.

5.8.1 The Problem with the Pentium 4

Before getting into the details of the IA-64 and Itanium 2, it is useful to review what is wrong with the Pentium 4 to see what problems Intel was trying to solve with the new architecture. The main fact of life that causes all the trouble is that IA-32 is an ancient ISA with all the wrong properties for current technology. It is a CISC ISA with variable-length instructions and a myriad of different formats that are hard to decode quickly on the fly. Current technology works best with RISC ISAs that have one instruction length and a fixed-length opcode that is easy to decode. The IA-32 instructions can be broken up into RISC-like micro-operations at execution time, but doing so requires hardware (chip area), takes time, and adds complexity to the design. That is strike one.

The IA-32 is also a two-address memory-oriented ISA. Most instructions reference memory, and most programmers and compilers think nothing of referencing memory all the time. Current technology favors load/store ISAs that only reference memory to get the operands into registers but otherwise perform all their calculations using three-address memory register instructions. And with CPU clock speeds going up much faster than memory speeds, the problem will get worse with time. That is strike two.

The IA-32 also has a small and irregular register set. Not only does this tie compilers in knots, but the small number of general-purpose registers (four or six, depending on how you count ESI and EDI) requires intermediate results to be spilled into memory all the time, generating extra memory references even when they are not logically needed. That is strike three. The IA-32 is out.

Now let us start the second inning. The small number of registers causes many dependences, especially unnecessary WAR dependences, because results have to go somewhere and there are no extra registers available. Getting around the lack of registers requires the implementation to do renaming internally—a terrible hack if ever there was one—to secret registers inside the reorder buffer. To avoid blocking on cache misses too often, instructions have to be executed out of order. However, the IA-32's semantics specify precise interrupts, so the out-of-order instructions have to be retired in order. All of these things require a lot of very complex hardware. Strike four.

Doing all this work quickly requires a deep pipeline. In turn, the deep pipeline means that instructions are entered into it take many cycles before they are finished. Consequently, very accurate branch prediction is essential to make sure the right instructions are being entered into the pipeline. Because a misprediction requires the pipeline to be flushed, at great cost, even a fairly low misprediction rate can cause a substantial performance degradation. Strike five.

To alleviate the problems with mispredictions, the processor has to do speculative execution, with all the problems that entails, especially when memory references on the wrong path cause an exception. Strike six.

We are not going to play the whole baseball game here, but it should be clear by now that there is a real problem. And we have not even mentioned the fact that IA-32's 32-bit addresses limit individual programs to 4 GB of memory, which is a growing problem on high-end servers. The EMT-64 solves this problem but not all the others.

All in all, the situation with IA-32 can be favorably compared to the state of celestial mechanics just prior to Copernicus. The then-current theory dominating astronomy was that the earth was fixed and motionless in space and that the planets moved in circles with epicycles around it. However, as observations got better and more deviations from this model could be clearly observed, epicycles were added to the epicycles until the whole model just collapsed from its internal complexity.

Intel is in the same pickle now. A huge fraction of all the transistors on the Pentium 4 are devoted to decomposing CISC instructions, figuring out what can be done in parallel, resolving conflicts, making predictions, repairing the consequences of incorrect predictions and other bookkeeping, leaving surprisingly few over for doing the real work the user asked for. The conclusion that Intel is being inexorably driven to is the only sane conclusion: junk the whole thing (IA-32) and start all over with a clean slate (IA-64). The EMT-64 provides some breathing room, but it really papers over the complexity issue.

5.8.2 The IA-64 Model: Explicitly Parallel Instruction Computing

The key idea behind the IA-64 is moving work from run time to compile time. On the Pentium 4, during execution the CPU reorders instructions, renames registers, schedules functional units, and does a lot of other work to determine how to keep all the hardware resources fully occupied. In the IA-64 model, the compiler figures out all these things in advance and produces a program that can be run as