

A Discussion of the Design Decisions Made in the Intel Itanium Processor

Edward Dale
Computer Architecture
Spring 2003

The state of 64-bit computing is at a crossroads right now. The big two consumer CPU manufacturers are offering major new designs. For the first time in more than 15 years, Intel is going to take a path will break backwards compatibility with its x86 instruction set. The new instruction set is called IA64 and the first generation of CPU to utilize it is the Itanium. In contrast, AMD's plan is keep backwards compatibility while allowing 64-bit programs to exist. Obviously, many choices were made in the creation of this new CPU and this paper seeks to explain some of the changes and impetus behind. First, the reasons for a 64-bit CPU will be discussed. The novel design decisions that Intel made will then be discussed, followed by the implications that these decisions have had on the new IA64 instruction set. Finally, the methods that Intel is using to allow x86 code to continue to be executed on the Itanium will be discussed.

64-bit computing is the natural progression of the industry. Fifteen years ago, the switch was made from 16 bits to 32 bits and now the switch is slowly being made to 64 bits. The switch is being made slowly because the Itanium has been in design since at least 1994 and in manufacture since 1998 and still has barely any market share. That market share has almost been exclusively in the server and high-end workstation markets. Partly responsible for this is the fact that these computers are the ones that are going to see the most benefit. Servers usually require a lot of memory, a fact that is corroborated by noticing that most servers now come with multiple gigabytes of memory, sometimes approaching the 64GB maximum of today's Xeon processors. Servers shipping with their maximum amount of memory already filled have little to no upgrade path, which is generally a bad thing. On a 64-bit architecture, there is an upper limit of 1.84×10^7 terabytes of addressable space, a limit that will not soon be reached. As mentioned, 64-bit computing is also useful for high-end workstations. Obviously, every facet of these workstations do not benefit from the extra bits. Indeed, only certain algorithms that can work in

parallel on sets of data will possibly see a benefit. Examples of such algorithms include searching for character in a string or 3d rendering. This type of processing is called SIMD (Single Instruction Multiple Data) and works by loading each character of a string into a 64-bit quad-word and processing it in parallel. As will be shown, the IA64 and MMX instruction sets have instructions that specifically address this requirement. Moving to 64-bits has a direct affect on these instructions because all of the sudden twice as much data can be processed at once. The last reason that moving to 64-bit computing is a good thing is because the move doubles the bandwidth between the register file and the L1 cache. This will be a big improvement to floating point performance because of the fact that floating point is already being implemented using 80-bit pathways inside the CPU [TURL03].

Intel has made many novel design decisions in the implementation of the first generation of Itanium processors. Most of these changes exemplify the EPIC architecture; a new acronym that stands for Explicitly Parallel Instruction set Computing. Some computer architects say that EPIC is basically just a rehashing of VLIW (Very Long Instruction Word), however EPIC includes a few additional enhancements. The benefits of the EPIC architecture include branch hints, cache hints, and predication. All of these are assisted by the design of the pipeline, which will be discussed first [TURL03].

At first glance, the pipeline of the Itanium is nothing special, 10 stages deep and 6 instructions wide. However, its key feature is on the fetch side. During the first three stages, the CPU is able to fetch up to 6 instructions per cycle and put them into a buffer for the rest of the pipeline. This buffer effectively decouples the fetch process from the rest of the pipeline. By decoupling the fetch cycle, the rest of the pipeline can chug away decoding and executing with the instructions coming from a seemingly bottomless store of fetched instructions. The ultimate

goal of this decoupling is to prevent pipeline bubbles from occurring because of branch and cache misses. This is accomplished by the aggressive prediction provided by the branch hints of the EPIC architecture. There is also special branch prediction hardware that serves as a go-between for the execution and fetch units. This hardware includes a hierarchy of individual branch predictors that progressively predict branches with better accuracy [SHAR00].

Branch prediction has always been an area of possible improvement for computer architecture. The Itanium has taken a major step forward in this area by taking a lot of the burden of prediction off of the hardware and placing it in the compiler's hands in the form of branch hints. Hints of branch outcomes can be specified either on the actual branch instruction or with the BRP branch prediction instruction. The compiler can mark a BRP as important, which will place that branch prediction in a set of registers called the Target Address Registers that will allow for single-cycle turnaround on certain branches like loops. This importance marker is the first step in the branch prediction hierarchy mentioned earlier. Later stages of the branch prediction hierarchy have the ability to dynamically adjust previous predictions to account for loop exits and other dynamic conditions. This is managed by keeping track of the loop count, so the hardware itself knows when the loop is going to end. This adjustment completely avoids the possible branch miss that occurs on loop exit. Branch hints can also be used to initiate instruction prefetching in cases where the compiler knows a basic block is next. In these cases, anywhere from 64 bytes to “a long sequential instruction stream” can be prefetched. The hardware must be given a certain number of cycles notice for this technique to be advantageous, however, a compiler knows the layout of the instructions and knows to only do this when a benefit can be had [SHAR00].

The organization of the Itanium's memory includes an increase to three levels of cache. Because memory is such a bottleneck for CPUs and because Intel is moving more logic into the compiler's domain, the IA64 instruction set includes the ability for the compiler to hint to the hardware the temporal locality of each memory access. The hardware uses these hints to determine the optimum allocation and replacement strategy for each cache level. The compiler can also specify that it intends to modify certain pieces of cache data, so that the data can be brought up the cache hierarchy and prepared for the modification. Each of the five locality hints have a different effect depending on the cache level of the data and whether the data is a floating point or integer value [SHAR00].

Predication is the updated, more extendable version of the condition codes seen in earlier instruction sets. The inherent problem with condition codes is that there has to be a separate instruction to handle each possible outcome. Predicates allow an instruction's execution to be contingent upon the return value of a previous instruction. The revolution that has occurred with the IA64 instruction set is that almost every instruction can be predicated. The buzz-phrase that is associated with this advancement is that it makes a control dependency (branches) become a data dependency (predicates). Predicates are implemented as six bits at the end of most instructions, each of which specify a certain dependency. Load, store, and other instructions can specify the predicate bits that that instruction fulfills. Because predication creates a data dependency, a predicated instruction can utilize forwarding and other hazard detection techniques to stall until the dependency is met. All possible measures have been taken to assure that stalls are the last choice, because if the pipeline is stalled all the time, then no benefit has been reaped over branch prediction [SHAR00].

One of the major selling points and buzzwords of Intel's new processor and instruction set is Instruction Level Parallelism (ILP). Utilizing the EPIC paradigm does a good job of furthering this goal by using branch prediction, cache prediction, and predication. ILP and the VLIW nature of the Itanium mean that when instructions are issued, they need to be issued in the EPIC version of a VLIW, an instruction bundle. Instruction bundles are simply combinations of three single instructions that are combined and issued to the CPU together. Because the Itanium is a six-issue processor, two bundles can be executed at once. Instructions fall into one of six categories: memory, integer, floating point, load, branch, and branch, hints. Different combinations of these instructions yield different bundles and different pairs of bundles will take up a varying amount of processor bandwidth. There are twenty-seven optimal pairs of bundles that take advantage of the full bandwidth of the processor. Obviously, it is these bundles that a compiler is going to target. If that is not possible, the compiler can issue a non-optimal pair of bundles, however, in some cases, it may be advantageous to execute a NOP for one of the instructions instead of suffer the possible consequences of issuing an instruction that does not fit in the nature of the parallelism of that bundle [SHAR00].

The effect of some of the design decisions on the instruction set has already been mentioned in passing. The BRP instruction is entirely new to the IA64 instruction set as are the branch hints tacked onto the end of the branch instructions. The x86 instruction set also has no concept of the locality hints that are part of memory instructions. Another big change that affects the entire instruction set is the addition of the predication bits. Predication also eliminates any instructions that used condition codes or similar x86 structures. A change that does not necessarily relate directly to the x86 instruction is the lack of support for SSE and MMX instructions. These instructions operate on multiple integers packed into two double words. This

concept has been integrated into the IA64 instruction set in the form of specific PACK and UNPACK instructions to put the data into parallel structures and instructions beginning with P to do parallel operations on the packed data. This could possibly see a step backwards considering that compilers have been playing catch-up for the past five years to fully utilize the MMX and SSE extensions. However, one would hope that the compilers are smart enough to translate MMX and SSE instructions to efficiently use the new packing of the Itanium [SHAR00].

One of the major barriers for any new CPU is the question of whether code from the previous CPU will be able to execute. Intel has circumvented this hurdle for the past fifteen years by remaining completely backwards compatible and just tacking on new instructions in addition to current ones. This has obviously led to a lot of cruft in the x86 instruction set. Because Intel has chosen to shed the shackles of x86, they have finally run into the barrier of no backwards compatibility. The current solution to this issue is full hardware emulation of the x86 instruction set in the CPU. There is dedicated x86 decode and control circuitry on the front-end of the CPU that steps in when the 32-bit code is encountered [SHAR00]. However, this has proven to be an insufficient solution, so much so that even Intel does not recommend using the Itanium if you will be using 32-bit programs for any appreciable amount of time. Intel has proposed a solution to this problem just in the past week. Their plan is to take a software approach instead of using dedicated circuitry. This new approach, named the IA-32 Execution Layer, is currently being integrated into popular OSes like Windows and Linux. Intel anticipates the 32-bit performance of the forthcoming 1.5Ghz Itanium 2 to be roughly that of a 1.5Ghz Xeon MP. There are numerous additional benefits to moving 32-bit support into a software solution. First, the software can more easily be adapted to future technology and even currently unsupported technology such as the SSE instructions the Itanium lacks. The Itanium could also

lose the hardware support it currently has and go strictly with the software support. A change such as this would not only free up space on the CPU, but could also allow Intel to change the Itanium's design to eliminate any design compromises they had to make for 32-bit support [SHAN03].

Computing is always advancing and always hitting barriers in its progress. The move to 64-bit computing is a barrier that much thought and effort has been invested in to pass. Soon, consumers will begin to see the fruits of this labor in the form of new 64-bit CPUs from AMD and Intel. Because computers have reached such dizzying speeds, every design decision that is made is done with years of research backing it up. By analyzing the motives and methods of these decisions, a wealth of information can be gleaned. The decision made by Intel to use an EPIC architecture with their new CPU brings to light many of the core principles of computer architecture. The level of parallelism inherent in the instruction set is a direct result of the design decisions made in the fetch cycle and the rest of the pipeline. The Itanium CPU and IA64 instruction set represent bold steps in a direction that heretofore Intel has been unwilling to take.

Bibliography

[SHAR00] Sharangpani, H., and Arora, K.. Itanium Processor Microarchitecture. IEEE Micro, Sep-Oct 2000.

[HUCK00] Huck, J. and others, Introducing the IA-64 Architecture. IEEE Micro, Sep-Oct 2000.

[CRAW00] Crawford, J., Introducing the Itanium Processors. IEEE Micro, Sep-Oct 2000.

[TURL03] Turley, Jim, 64-Bit CPUs: What You Need to Know. [web page] Feb 2003;
<http://www.extremetech.com/article2/0,3973,38732,00.asp> [Accessed 28 April 2003].

[SHAN03] Shankland, Stephen, Intel plans Itanium course correction. [web page] April 2003;
<http://news.com.com/2100-1006-997936.html> [Accessed 28 April 2003].