x86-64 Machine-Level Programming*

Randal E. Bryant
David R. O'Hallaron

September 8, 2008

Intel's IA32 instruction set architecture (ISA), colloquially known as "x86", is the dominant instruction format for the world's computers. IA32 is the platform of choice for most Windows, Linux, and, since 2006, even Macintosh computers. The ISA we use today was defined in 1985 with the introduction of the i386 microprocessor, extending the 16-bit instruction set defined by the original 8086 to 32 bits. Even though subsequent processor generations have introduced new instruction types and formats, many compilers, including GCC, have avoided using these features in the interest of maintaining backward compatibility.

A shift is underway to a 64-bit version of the Intel instruction set. Originally developed by Advanced Micro Devices (AMD) and named *x86-64*, it is now supported by high end processors from AMD (who now call it *AMD64*) and by Intel, who refer to it as *Intel64*. Most people still refer to it as "x86-64," and we follow this convention. (Some vendors have shortened this to simply "x64"). Newer versions of Linux and GCC support this extension. In making this switch, the developers of GCC saw an opportunity to also make use of some of the instruction-set features that had been added in more recent generations of IA32 processors.

This combination of new hardware and revised compiler makes x86-64 code substantially different in form and in performance than IA32 code. In creating the 64-bit extension, the AMD engineers also adopted some of the features found in reduced-instruction set computers (RISC) [7] that made them the favored targets for optimizing compilers. For example, there are now 16 general-purpose registers, rather than the performance-limiting eight of the original 8086. The developers of GCC were able to exploit these features, as well as those of more recent generations of the IA32 architecture, to obtain substantial performance improvements. For example, procedure parameters are now passed via registers rather than on the stack, greatly reducing the number of memory read and write operations.

This document serves as a supplement to Chapter 3 of *Computer Systems: A Programmer's Perspective* (CS:APP), describing some of the differences. We start with a brief history of how AMD and Intel arrived at x86-64, followed by a summary of the main features that distinguish x86-64 code from IA32 code, and then work our way through the individual features.

1 History and Motivation for x86-64

Over the more than twenty years since introduction of the i386, the capabilities of microprocessors have changed dramatically. In 1985, a fully configured, high-end desktop computer had around 1 megabyte of random-access memory (RAM) and 50 megabytes of disk storage. Microprocessor-based "workstation" systems were just becoming the machines of choice for computing and engineering professionals. A typical microprocessor had a 5-megahertz clock and ran around one million instructions per second. Nowadays, a typical high-end system has 2 gigabyte of RAM (40X increase), 1 terabyte of disk storage (20,000X increase), and a 4-gigahertz clock, running around 5 billion instructions per second (5000X increase). Microprocessor-based systems have become pervasive. Even today's supercomputers are based on harnessing the power of many microprocessors computing in parallel. Given these large quantitative improvements, it is remarkable that the world's computing base mostly runs code that is binary compatible with machines that existed over 20 years ago.

The 32-bit word size of the IA32 has become a major limitation in growing the capacity of microprocessors. Most significantly, the word size of a machine defines the range of virtual addresses that programs can use, giving a 4-gigabyte virtual address space in the case of 32 bits. It is now feasible to buy more than this amount of RAM for a machine, but the system cannot make effective use of it. For applications that involve manipulating large data sets, such as scientific computing, databases, and data mining, the 32-bit word size makes life difficult for programmers. They must write code using *out-of-core* algorithms[1], where the data reside on disk and are explicitly read into memory for processing.

Further progress in computing technology requires a shift to a larger word size. Following the tradition of growing word sizes by doubling, the next logical step is 64 bits. In fact, 64-bit machines have been available for some time. Digital Equipment Corporation introduced its Alpha processor in 1992, and it became a popular choice for high-end computing. Sun Microsystems introduced a 64-bit version of its SPARC architecture in 1995. At the time, however, Intel was not a serious contender for high-end computers, and so the company was under less pressure to switch to 64 bits.

Intel's first foray into 64-bit computers were the Itanium processors, based on the IA64 instruction set. Unlike Intel's historic strategy of maintaining backward compatibility as it introduced each new generation of microprocessor, IA64 is based on a radically new approach jointly developed with Hewlett-Packard. Its *Very Large Instruction Word* (VLIW) format packs multiple instructions into bundles, allowing higher degrees of parallel execution. Implementing IA64 proved to be very difficult, and so the first Itanium chips did not appear until 2001, and these did not achieve the expected level of performance on real applications. Although the performance of Itanium-based systems has improved, they have not captured a significant share of the computer market. Itanium machines can execute IA32 code in a compatibility mode but not with very good performance. Most users have preferred to make do with less expensive, and often faster, IA32-based systems.

Meanwhile, Intel's archrival, Advanced Micro Devices (AMD) saw an opportunity to exploit Intel's misstep with IA64. For years AMD had lagged just behind Intel in technology, and so they were relegated to competing with Intel on the basis of price. Typically, Intel would introduce a new microprocessor at a price premium. AMD would come along 6 to 12 months later and have to undercut Intel significantly to

[1] The physical memory of a machine is often referred to as *core memory*, dating to an era when each bit of a random-access memory was implemented with a magnetized ferrite core.

get any sales—a strategy that worked but yielded very low profits. In 2002, AMD introduced a 64-bit microprocessor based on its "x86-64" instruction set. As the name implies, x86-64 is an evolution of the Intel instruction set to 64 bits. It maintains full backward compatibility with IA32, but it adds new data formats, as well as other features that enable higher capacity and higher performance. With x86-64, AMD has sought to capture some of the high-end market that had historically belonged to Intel. AMD's recent generations of Opteron and Athlon 64 processors have indeed proved very successful as high performance machines. Most recently, AMD has renamed this instruction set *AMD64*, but "x86-64" persists as the favored name.

Intel realized that its strategy of a complete shift from IA32 to IA64 was not working, and so began supporting their own variant of x86-64 in 2004 with processors in the Pentium 4 Xeon line. Since they had already used the name "IA64" to refer to Itanium, they then faced a difficulty in finding their own name for this 64-bit extension. In the end, they decided to describe x86-64 as an enhancement to IA32, and so they referred to it as *IA32-EM64T* for "Enhanced Memory 64-bit Technology." In late 2006 they adopted the name *Intel64*.

On the compiler side, the developers of GCC steadfastly maintained binary compatibility with the i386, even as useful features were being added to the IA32 instruction set. The 1995 PentiumPro introduced a set of conditional move instructions that could greatly improve the performance of code involving conditional operations. More recent generations of Pentium processors introduced new floating point operations that could replace the rather awkward and quirky approach dating back to the 8087, the floating point coprocessor that accompanied the 8086 and, since the i486, has been included as part of the microprocessor itseulf. Switching to x86-64 as a target provided an opportunity for GCC to give up backward compatibility and instead exploit these newer features.

In this document, we use "IA32" to refer to the combination of hardware and GCC code found in traditional, 32-bit versions of Linux running on Intel-based machines. We use "x86-64" to refer to the hardware and code combination running on the newer 64-bit machines from AMD and Intel. In the Linux world, these two platforms are referred to as "i386" and "x86_64," respectively.

2 Finding Documentation

Both Intel and AMD provide extensive documentation on their processors. This includes general overviews of the assembly language programmer's view of the hardware [2, 4], as well as detailed references about the individual instructions [3, 5, 6]. The organization