



Porting to AMD64

Frequently Asked Questions

May 2003

© 2003 Advanced Micro Devices, Inc. All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel, or otherwise, to any intellectual property rights are granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Trademarks

AMD, the AMD Arrow logo, AMD Athlon, AMD Opteron, and combinations thereof and 3DNow! are trademarks of Advanced Micro Devices, Inc.

Windows is a registered trademark of Microsoft Corporation.

MMX is a trademark of Intel Corporation.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Frequently Asked Questions

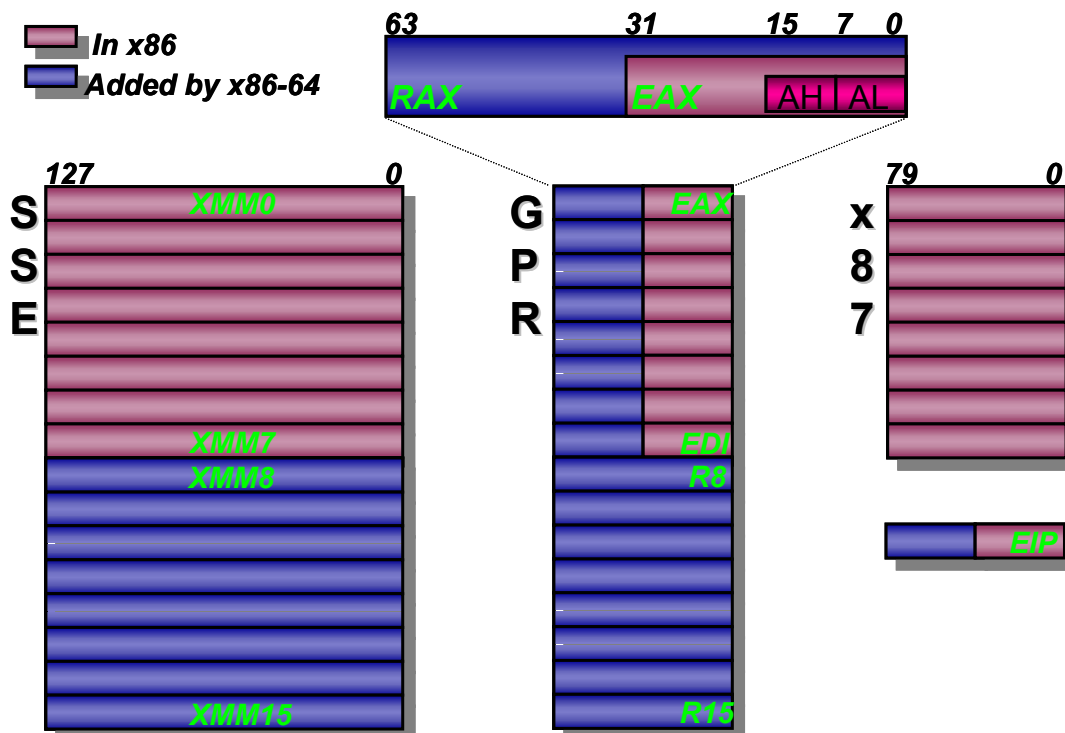
Background information

What is the difference between the AMD64 Instruction Set Architecture (ISA) and its predecessor, x86 architecture?

To create the AMD64 ISA, AMD started with the x86 architecture and extended it to 64 bits. These changes are simple but powerful:

- Widened General Purpose Registers (GPR) to 64 bits and increased the number of GPRs from 8 to 16
- All GPRs are addressable as 8, 16, 32, or 64 bits as needed
- Increased the number of 128-bit SSE registers from 8 to 16
- Widened the Extended Instruction Pointer (EIP) to 64 bits (now RIP)

The following illustration shows the programmer's model for the AMD64 ISA.



By minimizing the differences from x86 architecture, AMD designed the AMD64 ISA to be able to run virtually all 32-bit operating systems and application software without modification and at the same time allow for relatively easy porting to the power of 64-bit computing.

What are AMD's first processors using the AMD64 ISA?

The AMD64 technology is an AMD innovation that extends x86, the industry's most widely supported instruction set, and is designed to enable 64-bit computing while remaining compatible with the existing x86 software infrastructure and industry experience:

- AMD Athlon™ 64 is AMD's processor for desktop systems. Initially, we expect most of these systems to run 32-bit operating systems.
- AMD Opteron™ is AMD's processor for servers, supporting initially up to eight-way systems. It is expected that most of the larger servers will be running 64-bit operating systems.

All of these AMD64 processors are designed to support both 32-bit and 64-bit operating systems and applications.

32-bit compatibility

What are the compatibility issues of running a 32-bit application on a 32-bit operating system on AMD64 processors?

We've tested many 32-bit operating systems and 32-bit applications on all of our AMD64 reference systems and have not found any compatibility issues.

Are all 32-bit Windows® or Linux applications compatible with their 64-bit operating systems?

The AMD64 ISA is designed to impose no limitations on application compatibility, but that doesn't mean that every application that can run on a 32-bit operating system can also run on the 64-bit version. For example, Windows64 will not support 16-bit programs. AMD's 64-bit processors are designed to run any application supported by the operating system.

The following rare occurrences could cause compatibility problems with 32-bit applications:

- Applications that install device drivers may not work correctly if there is no 64-bit version of the required driver.

- Applications that attempt to share buffers containing pointers may not work; however, such programs do not follow normal conventions in either Linux or Windows and are not portable in general.

In general, what is the performance of 32-bit applications running under a 64-bit operating system?

64-bit operating systems tend to run faster and do a superior job of managing system resources

A 64-bit operating system can address much more physical memory than the 4GB limit of a 32-bit operating system. In 64-bit computer systems with large amounts of physical memory, 32-bit applications can have almost as much as 4GB of dedicated physical memory. Under most commercial 32-bit operating systems, the largest amount of physical memory available to an application is often 2GB, (the OS uses the remaining space of the total 4GB), which is further reduced by any other applications running. Having a larger amount of dedicated physical memory than would be possible under a 32-bit operating system can have a positive impact on performance.

Won't 32-bit applications need to go through a thunking or translation layer to make system calls to the 64-bit operating system, and won't this reduce performance?

A system call made to the 64-bit operating system by a 32-bit application will require translation of its arguments such as address pointers. This translation layer is integral to the 64-bit operating system. Some operating systems perform this translation through “thunking” while others perform this through “system call emulation”.

In the case of thunking, the thunking layer exists as a Dynamically Linked Library (DLL) that is integral to the operating system. The OS creates a separate 32-bit process to run each 32-bit application, and the thunking DLL resides within each of these 32-bit processes context. The 32-bit application is dynamically linked to the thunking layer. Each time the application executes a system call to the operating system the thunking layer is invoked and performs the following sequence of operations:

- Translates parameters if necessary
- Transfers control to (Calls) the 64-bit kernel
- Translates the results if necessary
- Returns the results to the application

The 64-bit AMD64 Linux and other UNIX operating systems use “system call emulation”. A system call on Linux involves transferring control to the kernel (operating system) with the system call's unique ID in a register. The kernel then branches to the proper routine based on the ID. There are 64-bit and 32-bit versions

of the important system calls, each has a unique ID. The 32-bit system call IDs are compatible with those of 32-bit x86 Linux kernels.

When the 32-bit application executes a system call to the kernel, it uses the ID of a 32-bit system call, and the kernel transfers control to the 32-bit version of the system call. In most cases, the 32-bit version of the system call will translate parameters if necessary and call the 64-bit version of the system call to do the actual work. The 64-bit system call will return back to the 32-bit version, which will translate results if necessary, and return to the 32-bit application.

There is some performance overhead associated with this translation each time a system-call occurs; however, the overhead is relatively small and typically short compared to the kernel's operation. In addition, most programs do a relatively small number of system calls compared to the amount of time the application is running. The translation of system calls is used in Windows today to allow 16-bit programs to run on 32-bit operating systems, and in the case of UltraSPARC and MIPS, it is used to allow 32-bit applications to run on 64-bit operating systems. These systems enjoy reasonable performance.

Degenerate cases exist where this thinking overhead is noticeable and these cases would be excellent candidates for porting.

What modifications do I need to make to a 32-bit Windows application so it can run on Microsoft's native 64-bit Windows OS?

None. 32-bit applications work fine under Microsoft's native 64-bit Windows OS. The infrastructure that allows this to happen is provided by Windows-on-Windows64 (WoW64). WoW64 provides the following environment.

- Windows32 ABI
- Use of 32-bit development tools
- Floating point calculations using x87
- Support for SSE, SSE2, MMX, and 3DNow!™ instructions

Why port to AMD64 architecture

If my 32-bit application runs on a 64-bit operating system, why should I bother to port it to 64 bits?

There may not be a need to port your application. We anticipate the majority of 32-bit applications may never be ported to 64 bits. The main reason to port to AMD64 is having an application that would benefit from the larger virtual and physical address space afforded by 64-bit processing. Such applications include, but are certainly not limited by, one or all of the following:

- Database access
- 3D graphics and animation

- Large-integer and floating-point calculations
- Oil and gas simulations
- Large CAD modeling systems

Also, certain numerical applications can benefit:

- Cryptography (large-integer math)
- Scientific computing

Applications with these kinds of requirements can show dramatic improvements in capabilities and performance, which can justify the cost of porting, testing, and maintaining a 64-bit application.

Porting code

What are the best practices for making code portable between 32-bit and 64-bit environments?

In the area of 32-bit and 64-bit portability it is most critical that software not presume anything about the size of an address or its relationship to the size of an int, long, etc.

Microsoft maintains some guidelines on their Web site:

<http://msdn.microsoft.com/> (search for “porting to 64 bit” and “porting to Win64”)

MigraTEC is also a good source of portability guidelines:

<http://www.migratec.com>

Are 32-bit device drivers compatible with 64-bit operating systems?

No. Only 64-bit drivers can be used with 64-bit operating systems.

How do 64-bit device drivers support hardware devices that are only capable of 32-bit PCI addressing?

A driver can allocate a buffer below the 32-bit address limit and direct the device to access that buffer. The driver then copies data to or from this buffer to the other software components that need this data anywhere in the 64-bit address space.

The 64-bit AMD64 Linux kernel provides another technique on AMD Athlon 64 and AMD Opteron systems: the I/O Memory Management Unit (IOMMU). This is essentially a generalization of the AGP Graphics Aperture translation mechanism to support other I/O Devices.

How is the AGP aperture mapped to support AGP devices that are only capable of 32-bit PCI addressing?

Regardless of the amount of DRAM in the system, the BIOS maps the AGP aperture just below the 4GB boundary to put it within 32-bit addressing capability. There should never be an addressing conflict between the AGP aperture and DRAM in a system with close to or more than 4GB of DRAM.

Are there any tools available to assist developers in porting source code from 32-bit environments to AMD64?

In the Windows environment, compiling applications with the `/Wp64` switch will give warnings about non-portable source code constructs, even when compiling for 32-bit targets. Cleaning your source code by using `/Wp64` now is an excellent first step in porting.

In the Linux environment, the following gcc switches can be used to catch portability issues:

```
-Werror -Wall -W -Wstrict-prototypes -Wmissing-prototypes -Wpointer-arith  
-Wreturn-type -Wcast-qual -Wwrite-strings -Wswitch -Wshadow -Wcast-align  
-Wuninitialized -ansi -pedantic -Wbad-function-cast -Wchar-subscripts  
-Winline -Wnested-externs -Wredundant-decl
```

MigraTEC's *64Express* is designed to automate 32-bit C/C++ source code migration to 64-bit Windows and Linux systems. With *64Express*, users efficiently and interactively direct the process, surpassing and eliminating conventional approaches that rely on compiler, lint and script methods.

<http://www.migratec.com>

Do all device drivers need to be ported to 64-bit for AMD64 versions of the Microsoft Windows and Linux operating systems?

Yes. A 64-bit operating system requires 64-bit drivers and kernel code. All kernel code and drivers need to be ported and recompiled for 64 bits.

Do all libraries need to be ported to 64-bit for AMD64 versions of the Microsoft Windows and Linux operating systems?

Only libraries that are used by 64-bit applications need to be ported.

Will DirectX clients need to be ported?

DirectX clients, like other 32-bit applications, should be able to operate properly as 32-bit applications running with Microsoft's native 64-bit Windows OS. In some cases

DirectX clients, especially ones with intense 3D graphics, can benefit from 64-bit performance, and they can be ported just like any other application.

Can you mix 32-bit and 64-bit code in the same process?

No.

What inter-process communication mechanisms will work between 32-bit processes and 64-bit processes?

IPC mechanisms directly supported by the operating system should support IPC between 32-bit and 64-bit processes.

What are the issues with porting an application from another 64-bit operating system?

If the code is designed for portability, there should be very few issues since the application will already be 64-bit clean. For example, when IBM ported the IBM DB2 Universal Database to AMD64 Linux, the entire process took only two days and most of that time was spent compiling the 1+ million lines of code.

What are the memory alignment requirements for code and data in each of these operating systems?

There are no alignment requirements. x86 architecture has never had any alignment requirements and the 64-bit extensions have not added any. However, the ABI for 64-bit software has been designed to improve performance by striving for natural alignment of most memory operands. Natural alignment means that 2-byte objects are stored on 2-byte boundaries, 4-byte objects on 4-byte boundaries, etc. The performance side-effects of poorly aligned operands can be large.

What are the consequences of unaligned memory access?

Performance degradation can occur when accessing misaligned memory operands.

Memory management

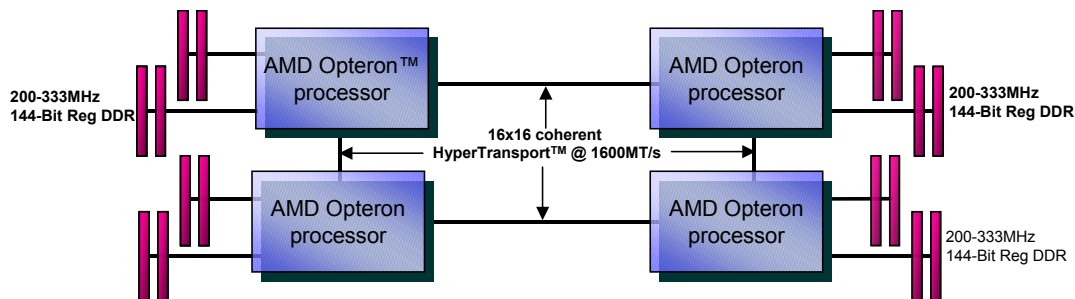
How does the on-board memory controller on the AMD Opteron™ and AMD Athlon™ 64 processors affect performance?

Each AMD64 processor includes a DDR memory controller that can control 4 or more DDR DIMMs. Memory performance is improved for the following reasons:

- The memory controller functions at CPU clock speeds and is not limited to the speed of the front-side bus.
- Best-case memory latency is significantly reduced when compared to non-integrated memory controller designs.

How do the on-board memory controllers affect memory in a multiprocessor system?

Since each processor has its own memory controller, in a multiprocessor system both memory bandwidth and total physical memory scales to the number of processors. The following illustration shows how AMD Opteron processors scale memory and memory bandwidth in a four-processor system.



AMD Opteron platform architecture does not require each processor to be associated with physical memory or that physical memory be evenly distributed among the processors. However, such a configuration may be desirable for optimal performance applications.

How is memory presented to the processor?

Each processor can have physical memory connected to it. After initialization, the memory map for the system presents all physical memory as a single contiguous extent.

This extent is formed in one of two ways. It is either:

- The concatenation of all processor memory. For example, all of processor 0's memory, followed by all of processor 1's memory, etc.
- The interleaving by page of all processor memory. For example, a page from processor 0's memory, followed by a page from processor 1's memory, etc.

The interleaved mechanism has better overall uniformity of latency but the concatenation mechanism is more usable for an operating system implementing memory affinity management.

After initialization, all memory appears equal to the casual programmer. No special addressing mechanisms are necessary to access physical memory attached to a processor other than the processor executing the memory reference instruction.

Is it necessary for the OS to have specific memory management considerations given the NUMA approach used by AMD Opteron systems?

Although not necessary, operating systems that provide careful memory and scheduling affinity improve overall memory latency and bandwidth and reduce traffic on the system's buses, which can have important performance improvement in some applications. Both Windows 64 and AMD64 Linux have some NUMA support for memory affinity and thread-biasing to enable optimal application performance considerations.

Processes and threads that use local data are advised to allocate (malloc) that local data space from within the thread or process that uses that data the most, to enable maximum benefit from the OS NUMA support.

Floating-point calculations in AMD64

Can x87 instructions still be used by 64-bit applications?

64-bit applications cannot use x87 instructions because 64-bit operating systems are not required to preserve the x87 stack across interrupts and context switches. AMD has gone to great lengths to ensure that SSE/SSE2 math library performance and accuracy exceeds that of the x87 instruction set. We anticipate no need to use x87 instructions in 64-bit applications.

Will there be any optimized 64-bit math libraries available?

Yes. On January 22, 2003, at the LinuxWorld Expo 2003, AMD announced that it is working with The Numerical Algorithms Group (NAG) to develop the AMD Core Math Library (ACML) to support AMD's upcoming AMD Opteron and AMD Athlon 64 processors.

ACML is planned to be composed of highly optimized numeric functions for mathematical, engineering, scientific and financial applications. The library, to be made available with both FORTRAN and C interfaces, is designed to be comprised of a full implementation of Level 1, 2 and 3 Basic Linear Algebra Subroutines (BLAS), Linear Algebra Package (LAPACK) as well as Fast Fourier Transforms (FFTs) in both single-, double-, single-complex and double-complex data types. ACML is designed to provide customers increased performance for computationally-intensive applications while taking full advantage of the performance benefits of the upcoming AMD Opteron and AMD Athlon 64 processors.

Porting to Microsoft's native 64-bit Windows OS

Which C/C++ compilers can I use for building Windows 64 applications?

Microsoft's Platform SDK and DDK are available now to developers of 64-bit applications and device drivers. These kits include 64-bit compilers and libraries. A 64-bit version of the Visual Studio development tools suite will be available in the future. Additionally, the Beta Release of the PGI Workstation 5.0 Fortran and C compilers for AMD Opteron is now available from The Portland Group Compiler Technology team of STMicroelectronics.

Which FORTRAN compilers can be used for building 64-bit Windows applications?

The Beta Release of the PGI Workstation 5.0 Fortran and C compilers for AMD Opteron is now available from The Portland Group Compiler Technology team of STMicroelectronics. Download the beta at: <http://www.pgroup.com/AMD64>.

Which compilers generate SSE/SSE2 code?

The AMD64 ABI specifies all floating-point operations performed by 64-bit applications utilize the SSE registers and the SSE/SSE2 instruction set, so all C, C++, FORTRAN and other compilers that are ABI-compliant generate SSE and SSE2 code for floating-point operations.

How can assembly language code be used with Microsoft's native 64-bit Windows operating system?

Microsoft's Visual Studio tools suite will no longer support inline assembly language for 64-bit processors. You can still link assembly-language subprograms into an application *after* you have assembled them with MASM. Intrinsics are also supported for SSE/SSE2 instructions and certain other functions.

Are there any performance profilers available for Microsoft's native 64-bit Windows OS?

A beta version of the AMD CodeAnalyst software supporting Windows 64 is available now. Download the beta at: <http://developer.amd.com>.

Are there any debug tools available for Windows?

Microsoft's Windows 64 operating system OS has the following debug tools:

- kd
- ntsd
- windbg

Compuware Corporation is developing SoftICE64 for AMD64. It will be available in beta form in 2003.

Are there resources on the Internet to help developers' port to Microsoft's native 64-bit operating system?

Microsoft has developer resources for porting code to 64-bit at:

<http://msdn.microsoft.com/> (search for "porting to 64-bit", "porting to Win64" or "AMD64")

The AMD64 web site (<http://www.x86-64.org/>) also has resources to help developers port their code to the AMD64 ISA.

Will an SDK and DDK be released for Microsoft's native 64-bit Windows OS for AMD64?

Microsoft's Platform SDK for 64-bit Windows currently provides the following:

- Cross-development command line tools
 - Tools are 32-bit binaries that produce 64-bit binaries
 - Can install on 64-bit Windows running on AMD64 (WoW64)
 - Can also be used remotely on Windows 32-bit platforms
- 64-bit "C" compiler
- 64-bit "MASM"
- 64-bit linker & librarian
- Disassembler
- Various C run-time and std DLLs.

- Driver build tools
- Native source level debug with ntsd (console-mode debugger)

For more information on the Microsoft SDK and DDK: <http://www.microsoft.com> (search for “Platform SDK” and “DDK”)

Porting to Linux 64

Are there any performance profilers available for Linux 64?

OProfile is a system-wide profiler for Linux systems, capable of profiling all running code at low overhead. OProfile is released under the GNU GPL. It consists of a kernel driver and a daemon for collecting sample data, and several post-profiling tools for turning data into information. For more information on OProfile: <http://oprofile.sourceforge.net/>

Where can I find the ABIs for AMD64 Linux?

The ABI for Linux 64 and other UNIX OS's is posted on the AMD64 web site (www.x86-64.org).

Which C/C++ compilers can I use for building 64-bit Linux applications?

gcc and assorted binutils are the *de facto* standards when developing for Linux. The Beta Release of the PGI Workstation 5.0 Fortran and C compilers for AMD Opteron is now available from The Portland Group Compiler Technology team of STMicroelectronics. Download the beta at: <http://www.pgroup.com/AMD64>.

Which FORTRAN compilers can be used for building 64-bit Linux applications?

The Beta Release of the PGI Workstation 5.0 Fortran and C compilers for AMD Opteron is now available from The Portland Group Compiler Technology team of STMicroelectronics. Download the beta at: <http://www.pgroup.com/AMD64>.

Which compilers generate SSE/SSE2 code?

The AMD64 ELF ABI specifies that all floating-point operations performed by 64-bit applications utilize the SSE register and the SSE/SSE2 instruction set, so all C, C++, FORTRAN, JAVA and other compilers that are ABI-compliant generate SSE and SSE2 code for floating-point operations.

How can assembly language code be used with Linux 64?

“GAS” assembly syntax for AMD64 can be found on www.x86-64.org. Assembly code can be compiled separately and linked in using “gas”. Also the 64-bit version of gcc supports inline assembly using the same method that the 32-bit version of gcc uses. Remember to use 64-bit AMD64 assembly instructions and not 32-bit x86 instructions when compiling the code for a 64-bit target.

Are there any debug tools available for AMD64 Linux?

Standard Linux debuggers are available for AMD64 Linux, including gdb, kdb, et.al.

Are there resources on the Internet to help developers' port to AMD64 Linux?

The AMD64 web site (<http://www.x86-64.org/>) has resources to help developers port their code to the AMD64 architecture.

Are there any specific issues for porting from 64-bit variations of UNIX to Linux 64?

Although there are a number of issues to be addressed when porting from various UNIX variants to Linux, none of them are specific to AMD64. Several excellent papers and books have been written on this topic.