X86-64

From OSDev Wiki

This article discusses **x86-64** CPUs (AMD64 and Intel's equivalent EM64T implementation). IA-64 (Itanium) is **really** a different beast and not addressed here.

Contents

Features

Long Mode

Long mode extends general registers to 64 bits (RAX, RBX, RIP, RSP, RFLAGS, etc), and adds an additional 8 integer registers (R8, R9, ..., R15) plus 8 more SSE registers (XMM8 to XMM15) to the CPU. Linear addresses are extended to 64 bit (however, a given CPU may implement less than this) and the physical address space is extended to 52 bits (a given CPU may implement less than this). In essence long mode adds another mode to the CPU.

Long mode does not support hardware task switching or virtual 8086 tasks. In long mode the current CS determines if the code currently running is 64 bit code (true long mode) or 32 bit code (compatibility mode), or even 16-bit protected mode code (still in compatibility mode). Using paging has become mandatory, and segmentation has been stripped down for performance reasons.

The first 64 bit CPUs from both Intel and AMD support 40 bit physical addresses and 48 bit linear addresses.

Segmentation in Long Mode

Segmentation in long mode functions with a flat model with the exception of two registers: FS and GS. Setting the base address for these two segment registers is possible via two specific Model Specific Register (MSR)s, FS.base (C000_0100h) and GS.base (C000_0101h).

Additionally there is a long mode specific instruction called SWAPGS, which swaps the contents of GS.base and another MSR called KernelGSBase (C000_0102h). This instruction is particularly useful for preserving kernel information for a specific logical processor core across context switches. **Note: This is an exchange operation**.

Further information

This feature overview is incomplete. Please see the [Wikipedia article on x86-64 (http://en.wikipedia.org/wiki/X86-64)] for more information.

Setting up

How do I detect if the CPU is 64 bits ?

After calling CPUID with EAX=0x80000001, all AMD64 compliant processors have the longmode-capable-bit turned on in the extended feature flags (bit 29) in EDX. There are also other bits required by long mode; you can check them out in the CPUID docs in the AMD general purpose instruction reference (http://support.amd.com/us/Processor_TechDocs/24594.pdf) (Link dead, the original author probably meant "AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions", found here: http://developer.amd.com/resources/documentation-articles/developer-guides-manuals/)

How do I enable Long Mode ?

The steps for enabling long mode are:

- Disable paging
- Set the PAE enable bit in CR4
- Load CR3 with the physical address of the PML4
- Enable long mode by setting the EFER.LME flag in MSR 0xC0000080
- Enable paging

Now the CPU will be in compatibility mode, and instructions are still 32-bit. To enter long mode, the D/B bit (bit 22, 2nd 32-bit value) of the GDT code segment must be clear (as it would be for a 16-bit code segment), and the L bit (bit 21, 2nd 32-bit value) of the GDT code segment must be set. Once that is done, the CPU is in 64-bit long mode.

Are there restrictions on 32-bit code running in Legacy Mode ?

x86-64 processors can operate in a legacy mode, they still start in real mode and 16 and 32 bit protected mode is still available (along with the associated Virtual 8086 mode). This means an x86 operating system, even DOS, will still run just fine. The only difference is that physical addresses can be up to 52 bits (or as many bits as implemented by the CPU) when PAE is used.

However, Virtual 8086 Mode does not exist in long/compatibility mode.

If you are running on a multi-processor system, you could send one processor a STARTUP IPI to a real mode memory address (see Intel MultiProcessor specification for more details) that loads a real mode program. The main problem with this approach is that it relies on multiple processors being present in the system.

Entering Long Mode directly

Protected mode must be entered before activating long mode. A minimal protected-mode environment must be established to allow long-mode initialization to take place. This environment must include the following:

- A protected-mode IDT for vectoring interrupts and exceptions to the appropriate handlers while in protected mode.
- The protected-mode interrupt and exception handlers referenced by the IDT.
- Gate descriptors for each handler must be loaded in the IDT.

> --*AMD64 docs, volume 2, section 14.4 (Enabling Protected Mode), 24593 Rev. 3.10 February 2005*

That being said, we have a thread where Brendan shows how to enable 64-bit long mode with no 32-bit IDT and no 32-bit segments - be assured, however, that any paging-related exception that occurs in long mode before you enable 64-bit IDT will cause the processor to reset due to a triple fault.

There is also an example of this implemented in a bootloader.

Notifying the BIOS

In order for the firmware built into the system to optimize itself for running in Long Mode, AMD recommends that the OS notify the BIOS about the intended target environment that the OS will be running in: 32-bit mode, 64-bit mode, or a mixture of both modes. This can be done by calling the BIOS interrupt 15h from Real Mode with AX set to 0xEC00, and BL set to 1 for 32-bit Protected Mode, 2 for 64-bit Long Mode, or 3 if both modes will be used.

64 bit Environment Models

There are three 64 bit programming models you need to consider: LP64, ILP64, LLP64. Each model has its own pitfalls. The I/L/P stand for Int, Long, Pointer, respectively; the 64 is the number of bits in each.

LP64 means Longs (and Long Longs) and Pointers are 64 bits wide, Ints are 32 bits wide. LLP64 means Long Longs and Pointers are 64 bits wide, Longs and Ints are 32 bit wide. ILP64 means Ints, Longs (and Long Longs) and Pointers are 64 bit wide.

Most *nixes use the LP64 model, Windows uses the LLP64 convention. ILP64 is used very rarely.

Data Types

This table lists the breakdown of sizes in the various programming models.

Datatype	LP64	ILP64	LLP64	ILP32	LP32
char	8	8	8	8	8
short	16	16	16	16	16

_int	32	--	32	--	--
int	32	64	32	32	16
long	64	64	32	32	32
long long	--	--	64	--	--
pointer	64	64	64	32	32

Models used by 64bit OSs

The following table lists what some current 64bit OS have as a programming model.

OS	Mode
Windows XP X64	LLP64
Linux	LP64
FreeBSD/OpenBSD	LP64
Solaris	LP64
DEC OSF/1 Alpha	LP64
SGI Irix	LP64
HP UX 11	LP64

See Also

Articles

- Intel EM64T
- Creating a 64-bit kernel
- X86-64 Instruction Encoding
- Setting up long mode

Wikipedia

- AMD64
- 64-bit

External Links

- Porting to AMD64: FAQ (http://www.amd.com/us-en/assets/content_type/DownloadableAssets/dwamd_AMD64_Porting_FAQ.pdf)
- AMD64 Information (http://www.amdboard.com/hammerspecial.html)
- x86-64 ABI and assembly guide (http://www.x86-64.org/documentation.html)
- ELF-64 Object File Format (direct PDF link) (http://downloads.openwatcom.org/ftp/devel/docs/elf-64-gen.pdf)
- StackOverflow x86_64 register assignment (http://stackoverflow.com/questions/1753602/registers-for-x86-64-processors)

Retrieved from "http://wiki.osdev.org/index.php?title=X86-64&oldid=17407"

Categories: X86 CPU | Operating Modes

- This page was last modified on 1 January 2015, at 12:55.
- This page has been accessed 110,459 times.