Continue

I have just noticed majority of my previous notifications is not sending out especially on those using notify.notify. I have just upgraded to 2022.3.5 from 2021.8.3, but I realise this issue is already there before i upgraded. So i went to developer tools and use the service tab to send out notify.notify. Even using the GUI. It seems not working. None of my IOS device received any notifications. However, when i choose notify.admin_group, it does works on those designated IOS device including iPhone & ipad. But if i use notify.mobile_apps_xxxxx, it only works on ipad but not on the iphone. This seems to be a major issue for me as i see the notification functions as a basic important functions for this home automations to work. All my IOS has been upgraded to the latest companion apps. Appreciate any advices notify.notify only works if you haven’t named a notification service. You are much better off using named notify services. e.g. notify.mobile_name_here or notify.group_name_here. Who will notify.notify notify? Configuration Just in case anyone comes across this post and thinks this is useful information, I’ve been doing a little research and this is what I’ve found: notify.notify will notify one of the platforms (and only one) without a “name” attribute. The thing is, when calling the service “notify.xxx” you are calling the notify platform and the “xxx” notification integration through it’s name. And here comes the confusion (and I think it’s not an ideal design), the name’s attribute default value is “notify” … Thanks for your response and advice. I created a separate notification group and changed the notify.notify to a notify.group_xxx and it works again. Don’t think the notify.notify can be used. I am trying to get notify to work for the first time and can’t seem to get it right. I have tried notify.notify and notify. and nothing seems to work. However the same yaml works in dev tools. I can’t think of anything else to try. The notify I am trying to fire is from keymaster. I had difficulty following your fix since this is new to me. I see nothing in the logs at all. Any ideas? Share your configuration for the notify action. I have two scripts due to conflicting documentation on keymaster. Neither works. Btw, I also tried notify.my_phone_entity Also did not work. avgjoeCO: notify.my_phone_entity Does this exist in Developer Tools → States? Also maybe ask here: KeyMaster Z-Wave lock manager and scheduler I know nothing about that integration. It was a bug in keymaster. They are working on it. This isn’t working for for me, suddenly none of my notify services are showing. Just simple email notify, all gone. Got mine working again, some reason the SMTP settings changed and made the service not appear Home Assistant has a very powerful notification system whereby it is possible to produce many different types of custom notification. Since the release of version 2 of the Home Assistant mobile app, support for custom push notifications has been added. Awesome! Like with other notifications, this type of notification is accessible by using the Home Assistant notify service. In this tutorial we will learn how to use the notify.mobile_app service and cover a basic example automation to trigger it. You should have Home Assistant installed and running. A basic understanding of YAML would also be highly beneficial. If you are not yet familiar with automation or YAML then I would recommend reading the tutorials about automation and scripts first. You will also need the Home Assistant app installed on at least one device. The Home Assistant Companion App can be downloaded from the iOS App Store or Play Store. You will also either need the file editor add-on or be familiar with editing the files using your favorite text editor. If you are not familiar with either you may want to check out my tutorial on automation or scripts first. You need to be running Home Assistant version 0.95 or newer, I would recommend just updating to the latest version. First we need to make sure that the default_config integration is added to our configuration.yaml file. If not then you can go ahead and add it. default_config: If there is some reason that you have removed it, you can instead enter the following. mobile_app: discovery: So long as you have either of these options within your configuration.yaml file, the companion app will automatically configure itself. It is important that you enable notifications during the installation. If you have already installed the companion app and did not enable notifications during the installation, you may need to follow the steps in the troubleshooting section. You can utilise push notifications anywhere that you would normally use the notify service in Home Assistant. For this example we will setup a basic automation to test and demonstrate the functionality. As we are creating a new automation, we will be adding the code to our automation integration within the configuration.yaml file. automation: Alternatively if you are using automation.yaml generated by the automation GUI or you already have automation in your configuration.yaml file, simply add the code beneath the existing code. Next we need to add a trigger for our automation. For this example we will create a simple notification that informs us when a lamp is switched on. automation: - alias: 'tree lamp notification' trigger: We will use the state platform in order to trigger the automation when the state of a light entity changes to ‘on,’ in this case my tree lamp. You should of course choose your own light entity. automation: - alias: 'tree lamp notification' trigger: platform: state entity_id: light.treelamp to: 'on' Now that we have an automation setup with our desired trigger, we can create an action that sends the push notification to our mobile device. We will use the service notify.mobile_app_ where will be the name of your mobile device integration in Home Assistant. In my case this is simply notify.mobile_app_iphone. automation: - alias: 'tree lamp notification' trigger: platform: state entity_id: light.treelamp to: 'on' action: - service: notify.mobile_app_iphone data: Now we can add our notification title and message as desired. automation: - alias: 'tree lamp notification' trigger: platform: state entity_id: light.treelamp to: 'on' action: - service: notify.mobile_app_iphone data: title: "Lights" message: "The tree lamp is on" When the notification fires, it will look something like this. If you are using an iOS device it is also possible to add a subtitle, which appears below the main title and above the message. automation: - alias: 'tree lamp notification' trigger: platform: state entity_id: light.treelamp to: 'on' action: - service: notify.mobile_app_iphone data: title: "Lights" message: "The tree lamp is on" data: subtitle: "Lamps" It is also possible to group notifications if you are using iOS 12 or newer. All notifications that share the same thread-id will be grouped in the notifications center. automation: - alias: 'tree lamp notification' trigger: platform: state entity_id: light.treelamp to: 'on' action: - service: notify.mobile_app_iphone data: title: "Lights" message: "The tree lamp is on" data: subtitle: "Lamps" push: thread-id: "lights-notification-group" When I first setup notifications I have some trouble getting them to work. If you had already installed the Home Assistant app prior to reading this tutorial and notifications are not working, you may need to carry out the following steps. Delete the Home Assistant app Remove the mobile device integration in Home Assistant from configuration > integrations Ensure that default_config: is present in the configuration.yaml file Restart Home Assistant from configuration > server controls Re-install the Home Assistant app That’s it! You should now be able to set up some awesome push notifications! In this tutorial we looked at a basic example of how a push notification can be sent to a mobile device using an automation. The integration works very well and makes for yet another superb Home Assistant feature. You should now have the basic knowledge required to add notifications to more useful applications, such as my easy Home Assistant DIY thermostat. Also check out my tutorial on push notifications and actionable buttons to add a button to your notification! Thanks for taking the time to read the tutorial and for visiting my site! Be sure to check out some of my other awesome Home Assistant tutorials! Thanks so much for visiting my site! If this article helped you achieve your goal and you want to say thanks, you can now support my work by buying me a coffee. I promise I won't spend it on beer instead... ☺ What version of Home Assistant Core has the issue? core-2025.4.4 What was the last working version of Home Assistant Core? No response What type of installation are you running? Home Assistant OS Integration causing the issue HTML5 Link to integration documentation on our website Diagnostics information No response Example YAML snippet Anything in the logs that might be useful for us? Additional information No response Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. 64-bit version of x86 architecture "Intel 64" redirects here. For the Intel 64-bit architecture in Itanium chips, see IA-64. "x64" redirects here. For the New York City bus route, see X64 (New York City bus). AMD Opteron, the first CPU to introduce the x86-64 extensions in April 2003 The five-volume set of the x86-64 Architecture Programmer's Manual, as published and distributed by AMD in 2002 x86-64 (also known as x64, x86_64, AMD64, and Intel 64)[note 1] is a 64-bit extension of the x86 instruction set. It was announced in 1999 and first available in the AMD Opteron family in 2003. It introduces two new operating modes: 64-bit mode and compatibility mode, along with a new four-level paging mechanism. In 64-bit mode, x86-64 supports significantly larger amounts of virtual memory and physical memory compared to its 32-bit predecessors, allowing programs to utilize more memory for data storage. The architecture expands the number of general-purpose registers from 8 to 16, all fully general-purpose, and extends their width to 64 bits. Floating-point arithmetic is supported through mandatory SSE2 instructions in 64-bit mode. While the older x87 FPU and MMX registers are still available, they are generally superseded by a set of sixteen 128-bit vector registers (XMM registers). Each of these vector registers can store one or two double-precision floating-point numbers, up to four single-precision floating-point numbers, or various integer formats. In 64-bit mode, instructions are modified to support 64-bit operands and 64-bit addressing mode. The x86-64 architecture defines a compatibility mode that allows 16-bit and 32-bit user applications to run unmodified alongside 64-bit applications, provided the 64-bit operating system supports them.[11][note 2] Since the full x86-32 instruction sets remain implemented in hardware without the need for emulation, these older executables can run with little or no performance penalty,[13] while newer or modified applications can take advantage of new features of the processor design to achieve performance improvements. Also, processors supporting x86-64 still power on in real mode to maintain backward compatibility with the original 8086 processor, as has been the case with x86 processors since the introduction of protected mode with the 80286. The original specification, created by AMD and released in 2000, has been implemented by AMD, Intel, and VIA. The AMD K8 microarchitecture, in the Opteron and Athlon 64 processors, was the first to implement it. This was the first significant addition to the x86 architecture designed by a company other than Intel. Intel was forced to follow suit and introduced a modified NetBurst family which was software-compatible with AMD's specification. VIA Technologies introduced x86-64 in their VIA Isaiah architecture, with the VIA Nano. The x86-64 architecture was quickly adopted for desktop and laptop personal computers and servers which were commonly configured for 16 GiB (gibibytes) of memory or more. It has effectively replaced the discontinued Intel Itanium architecture (formerly IA-64), which was originally intended to replace the x86 architecture. x86-64 and Itanium are not compatible on the native instruction set level, and operating systems and applications compiled for one architecture cannot be run on the other natively. AMD64 logo AMD64 (also variously referred to by AMD in their literature and documentation as "AMD 64-bit Technology" and "AMD x86-64 Architecture") was created as an alternative to the radically different IA-64 architecture designed by Intel and Hewlett-Packard, which was backward-incompatible with IA-32, the 32-bit version of the x86 architecture. AMD originally announced AMD64 in 1999[14] with a full specification available in August 2000.[15] As AMD was never invited to be a contributing party for the IA-64 architecture and any kind of licensing seemed unlikely, the AMD64 architecture was positioned by AMD from the beginning as an evolutionary way to add 64-bit computing capabilities to the existing x86 architecture while supporting legacy 32-bit x86 code, as opposed to Intel's approach of creating an entirely new, completely x86-incompatible 64-bit architecture with IA-64. The first AMD64-based processor, the Opteron, was released in April 2003. AMD's processors implementing the AMD64 architecture include Opteron, Athlon 64, Athlon 64 X2, Athlon 64 FX, Athlon II (followed by "X2", "X3", or "X4" to indicate the number of cores, and XLT models), Turion 64, Turion 64 X2, Sempron ("Palermo" E6 stepping and all "Manila" models), Phenom (followed by "X3" or "X4" to indicate the number of cores), Phenom II (followed by "X2", "X3", "X4" or "X6" to indicate the number of cores), FX, Fusion/APU and Ryzen/Epyc. The primary defining characteristic of AMD64 is the availability of 64-bit general-purpose processor registers (for example, rax), 64-bit integer arithmetic and logical operations, and 64-bit virtual addresses.[16] The designers took the opportunity to make other improvements as well. Notable changes in the 64-bit extensions include: 64-bit integer capability All general-purpose registers (GPRs) are expanded from 32 bits to 64 bits, and all arithmetic and logical operations, memory-to-register and register-to-memory operations, etc., can operate directly on 64-bit integers. Pushes and pops on the stack default to 8-byte strides, and pointers are 8 bytes wide. Additional registers In addition to increasing the size of the general-purpose registers, the number of named general-purpose registers is increased from eight (i.e. eax, ebx, ecx, edx, esi, edi, esp, ebp) in x86 to 16 (i.e. rax, rbx, rcx, rdx, rsi, rdi, rsp, rbp, r8, r9, r10, r11, r12, r13, r14, r15). It is therefore possible to keep more local variables in registers rather than on the stack, and to let registers hold frequently accessed constants; arguments for small and fast subroutines may also be passed in registers to a greater extent. AMD64 still has fewer registers than many RISC instruction sets (e.g. Power ISA has 32 GPRs; 64-bit ARM, RISC-V I, SPARC, Alpha, MIPS, and PA-RISC have 31) or VLIW-like machines such as the IA-64 (which has 128 registers). However, an AMD64 implementation may have far more internal registers than the number of architectural registers exposed by the instruction set (see register renaming). (For example, AMD Zen cores have 168 64-bit integer and 160 128-bit vector floating-point physical internal registers.) Additional XMM (SSE) registers Similarly, the number of 128-bit XMM registers (used for Streaming SIMD instructions) is also increased from 8 to 16. The traditional x87 FPU register stack is not included in the register file size extension in 64-bit mode, compared with the XMM registers used by SSE2, which did get extended. The x87 register stack is not a simple register file although it does allow direct access to individual registers by low cost exchange operations. Larger virtual address space The AMD64 architecture defines a 64-bit virtual address format, of which the low-order 48 bits are used in current implementations.[11]:120 This allows up to 256 TiB (248 bytes) of virtual address space. The architecture definition allows this limit to be raised in future implementations to the full 64 bits,[11]:2:3:13:117:120 extending the virtual address space to 16 EiB (264 bytes).[17] This is compared to just 4 GiB (232 bytes) for the x86.[18] This means that very large files can be operated on by mapping the entire file into the process's address space (which is often much faster than working with file read/write calls), rather than having to map regions of the file into and out of the address space. Larger physical address space The original implementation of the AMD64 architecture implemented 40-bit physical addresses and so could address up to 1 TiB (240 bytes) of RAM.[11]:24 Current implementations of the AMD64 architecture (starting from AMD 10h microarchitecture) extend this to 48-bit physical addresses[19] and therefore can address up to 256 TiB (248 bytes) of RAM. The architecture permits extending this to 52 bits in the future[11]:24[20] (limited by the page table entry format);[11]:131 this would allow addressing of up to 4 PiB of RAM. For comparison, 32-bit x86 processors are limited to 64 GiB of RAM in Physical Address Extension (PAE) mode,[21] or 4 GiB of RAM without PAE mode.[11]:4 Larger physical address space in legacy mode When operating in legacy mode the AMD64 architecture supports Physical Address Extension (PAE) mode, as do most current x86 processors, but AMD64 extends PAE from 36 bits to an architectural limit of 52 bits of physical address. Any implementation, therefore, allows the same physical address limit as under long mode.[11]:24 Instruction pointer relative data access Instructions can now reference data relative to the instruction pointer (RIP register). This makes position-independent code, as is often used in shared libraries and code loaded at run time, more efficient. SSE instructions The original AMD64 architecture adopted Intel's SSE and SSE2 as core instructions. These instruction sets provide a vector supplement to the scalar x87 FPU, for the single-precision and double-precision data types. SSE2 also offers integer vector operations, for data types ranging from 8bit to 64bit precision. This makes the vector capabilities of the architecture on par with those of the most advanced x86 processors of its time. These instructions can also be used in 32-bit mode. The proliferation of 64-bit processors has made these vector capabilities ubiquitous in home computers, allowing the improvement of the standards of 32-bit applications. The 32-bit edition of Windows 8, for example, requires the presence of SSE2 instructions.[22] SSE3 instructions and later Streaming SIMD Extensions instruction sets are not standard features of the architecture. No-Execute bit The No-Execute bit or NX bit (bit 63 of the page table entry) allows the operating system to specify which pages of virtual address space can contain executable code and which cannot. An attempt to execute code from a page tagged "no execute" will result in a memory access violation, similar to an attempt to write to a read-only page. This should make it more difficult for malicious code to take control of the system via "buffer overrun" or "unchecked buffer" attacks. A similar feature has been available on x86 processors since the 80286 as an attribute of segment descriptors; however, this works only on an entire segment at a time. Segmented addressing has long been considered an obsolete mode of operation, and all current PC operating systems in effect bypass it, setting all segments to a base address of zero and (in their 32-bit implementation) a size of 4 GiB. AMD was the first x86-family vendor to implement no-execute in linear addressing mode. The feature is also available in legacy mode on AMD64 processors, and recent Intel x86 processors, when PAE is used. Removal of older features A few "system programming" features of the x86 architecture were either unused or underused in modern operating systems and are either not available on AMD64 in long (64-bit and compatibility) mode, or exist only in limited form. These include segmented addressing (although the FS and GS segments are retained in vestigial form for use as extra-base pointers to operating system structures),[11]:70 the task state switch mechanism, and virtual 8086 mode. These features remain fully implemented in "legacy mode", allowing these processors to run 32-bit and 16-bit operating systems without modifications. Some instructions that proved to be rarely useful are not supported in 64-bit mode, including saving/restoring of segment registers on the stack, saving/restoring of all registers (PUSHA/POPA), decimal arithmetic, BOUND and INTO instructions, and "far" jumps and calls with immediate operands. Canonical address space implementations (diagrams not to scale)Current 48-bit implementation57-bit implementation64-bit implementation Although virtual addresses are 64 bits wide in 64-bit mode, current implementations (and all chips that are known to be in the planning stages) do not allow the entire virtual address space of 264 bytes (16 EiB) to be used. This would be approximately four billion times the size of the virtual address space on 32-bit machines. Most operating systems and applications will not need such a large address space for the foreseeable future, so implementing such wide virtual addresses would simply increase the complexity and cost of address translation with no real benefit. AMD, therefore, decided that, in the first implementations of the architecture, only the least significant 48 bits of a virtual address would actually be used in address translation (page table lookup).[11]:120 In addition, the AMD specification requires that the most significant 16 bits of any virtual address, bits 48 through 63, must be copies of bit 47 (in a manner akin to sign extension). If this requirement is not met, the processor will raise an exception.[11]:131 Addresses complying with this rule are referred to as "canonical form."[11]:130 Canonical form addresses run from 0 through 00007FFF'FFFFFFFF, and from FFFF8000'00000000 through FFFFFFFF'FFFFFFFF, for a total of 256 TiB of usable virtual address space. This is still 65,536 times larger than the virtual 4 GiB address space of 32-bit machines. This feature eases later scalability to true 64-bit addressing. Many operating systems (including, but not limited to, the Windows NT family) take the higher-addressed half of the address space (named kernel space) for themselves and leave the lower-addressed half (user space) for application code, user mode stacks, heaps, and other data regions.[23] The "canonical address" design ensures that every AMD64 compliant implementation has, in effect, two memory halves: the lower half starts at 00000000'00000000 and "grows upwards" as more virtual address bits become available, while the higher half is "docked" to the top of the address space and grows downwards. Also, enforcing the "canonical form" of addresses by checking the unused address bits prevents their use by the operating system in tagged pointers as flags, privilege markers, etc., as such use could become problematic when the architecture is extended to implement more virtual address bits. The first versions of Windows for x64 did not even use the full 256 TiB; they were restricted to just 8 TiB of user space and 8 TiB of kernel space.[23] Windows did not support the entire 48-bit address space until Windows 8.1, which was released in October 2013.[23] The 64-bit addressing mode ("long mode") is a superset of Physical Address Extensions (PAE); because of this, page sizes may be 4 KiB (212 bytes) or 2 MiB (221 bytes).[11]:120 Long mode also supports page sizes of 1 GiB (230 bytes).[11]:120 Rather than the three-level page table system used by systems in PAE mode, systems running in long mode use four levels of page table: PAE's Page-Directory Pointer Table is extended from four entries to 512, and an additional Page-Map Level 4 (PML4) Table is added, containing 512 entries in 48-bit implementations.[11]:131 A full mapping hierarchy of 4 KiB pages for the whole 48-bit space would take a bit more than 512 GiB of memory (about 0.195% of the 256 TiB virtual space). 64 bit page table entry Bits: 63 62 ... 52 51 ... 32 Content: NX reserved Bit 51...32 of base address Bits: 31 ... 12 11 ... 9 8 7 6 5 4 3 2 1 0 Content: Bit 31...12 of base address ign. G PAT D A PCD PWT U/S R/W P Intel has implemented a scheme with a 5-level page table, which allows Intel 64 processors to support 57-bit addresses, and in turn, a 128 PiB virtual address space.[24] Further extensions may allow full 64-bit virtual address space and physical memory with 12-bit page table descriptors and 16- or 21-bit memory offsets for 64 KiB and 2 MiB page allocation sizes; the page table entry would be expanded to 128 bits to support additional hardware flags for page size and virtual address space size.[25] The operating system can also limit the virtual address space. Details, where applicable, are given in the "Operating system compatibility and characteristics" section. Current AMD64 processors support a physical address space of up to 248 bytes of RAM, or 256 TiB.[19] However, as of 2020[update], there were no known x86-64 motherboards that support 256 TiB of RAM.[26][27][28][29][failed verification] The operating system may place additional limits on the amount of RAM that is usable or supported. Details on this point are given in the "Operating system compatibility and characteristics" section of this article. The architecture has two primary modes of operation: long mode and legacy mode. Operating Operating system required Type of code being run Size (in bits) No. of general-purpose registers Mode Sub-mode Addresses Operands (default in italics) Long mode 64-bit mode 64-bit OS, 64-bit UEFI firmware, or the previous two interacting via a 64-bit firmware's UEFI interface 64-bit 64 8, 16, 32, 64 16 Compatibility mode Bootloader or 64-bit OS 32-bit 32 8, 16, 32 8 16-bit protected mode 16 8, 16, 32 8 Legacy mode Protected mode Bootloader, 32-bit OS, 32-bit UEFI firmware, or the latter two interacting via the firmware's UEFI interface 32-bit 32 8, 16, 32 8 16-bit protected mode OS 16-bit protected mode 16 8, 16, 32[m 1] 8 Virtual 8086 mode 16-bit protected mode or 32-bit OS subset of real mode 16 8, 16, 32[m 1] 8 Unreal mode Bootloader or real mode OS real mode 16, 20, 32 8, 16, 32[m 1] 8 Real mode Bootloader, real mode OS, or any OS interfacing with a firmware's BIOS interface[30] real mode 16, 20, 21 8, 16, 32[m 1] 8 ^ a b c d Note that 16-bit code written for the 80286 and below does not use 32-bit operand instructions. Code written for the 80386 and above can use the operand-size override prefix (0x66). Normally this prefix is used by protected and long mode code for the purpose of using 16-bit operands, as that code would be running in a code segment with a default operand size of 32 bits. In real mode, the default operand size is 16 bits, so the 0x66 prefix is interpreted differently, changing operand size to 32 bits. State diagram of the x86-64 operating modes Main article: Long mode Long mode is the architecture's intended primary mode of operation; it is a combination of the processor's native 64-bit mode and a combined 32-bit and 16-bit compatibility mode. It is used by 64-bit operating systems. Under a 64-bit operating system, 64-bit programs run under 64-bit mode, and 32-bit and 16-bit protected mode applications (that do not need to use either real mode or virtual 8086 mode in order to execute at any time) run under compatibility mode. Real-mode programs and programs that use virtual 8086 mode at any time cannot be run in long mode unless those modes are emulated in software.[11]:11 However, such programs may be started from an operating system running in long mode on processors supporting VT-x or AMD-V by creating a virtual processor running in the desired mode. Since the basic instruction set is the same, there is almost no performance penalty for executing protected mode x86 code. This is unlike Intel's IA-64, where differences in the underlying instruction set mean that running 32-bit code must be done either in emulation of x86 (making the process slower) or with a dedicated x86 coprocessor. However, on the x86-64 platform, many x86 applications could benefit from a 64-bit recompile, due to the additional registers in 64-bit code and guaranteed SSE2-based FPU support, which a compiler can use for optimization. However, applications that regularly handle integers wider than 32 bits, such as cryptographic algorithms, will need a rewrite of the code handling the huge integers in order to take advantage of the 64-bit registers. Legacy mode is the mode that the processor is in when it is not in long mode.[11]:14 In this mode, the processor acts like an older x86 processor, and only 16-bit and 32-bit code can be executed. Legacy mode allows for a maximum of 32 bit virtual addressing which limits the virtual address space to 4 GiB.[11]:14:24:118 64-bit programs cannot be run from legacy mode. Protected mode is made into a submode of legacy mode.[11]:14 It is the submode that 32-bit operating systems and 16-bit protected mode operating systems operate in when running on an x86-64 CPU.[11]:14 Real mode is the initial mode of operation when the processor is initialized, and is a submode of legacy mode. It is backwards compatible with the original Intel 8086 and Intel 8088 processors. Real mode is primarily used today by operating system bootloaders, which are required by the architecture to configure virtual memory details before transitioning to higher modes. This mode is also used by any operating system that needs to communicate with the system firmware with a traditional BIOS-style interface.[30] Intel 64 is Intel's implementation of x86-64, used and implemented in various processors made by Intel. Historically, AMD has developed and produced processors with instruction sets patterned after Intel's original designs, but with x86-64, roles were reversed: Intel found itself in the position of adopting the ISA that AMD created as an extension to Intel's own x86 processor line. Intel's project was originally codenamed Yamhill[31] (after the Yamhill River in Oregon's Willamette Valley). After several years of denying its existence, Intel announced at the February 2004 IDF that the project was indeed underway. Intel's chairman at the time, Craig Barrett, admitted that this was one of their worst-kept secrets.[32][33] Intel's name for this instruction set has changed several times. The name used at the IDF was CT[34] (presumably[original research?] for Clackamas Technology, another codename from an Oregon river); within weeks they began referring to it as IA-32e (for IA-32 extensions) and in March 2004 unveiled the "official" name EM64T (Extended Memory 64 Technology). In late 2006 Intel began instead using the name Intel 64 for its implementation, paralleling AMD's use of the name AMD64.[35] The first processor to implement Intel 64 was the multi-socket processor Xeon code-named Nocona in June 2004. In contrast, the initial Prescott chips (February 2004) did not enable this feature. Intel subsequently began selling Intel 64-enabled Pentium 4s using the E0 revision of the Prescott core, being sold on the OEM market as the Pentium 4, model F. The E0 revision also adds eXecute Disable (XD) (Intel's name for the NX bit) to Intel 64, and has been included in then current Xeon code-named Irwindale. Intel's official launch of Intel 64 (under the name EM64T at that time) in mainstream desktop processors was the N0 stepping Prescott-2M. The first Intel mobile processor implementing Intel 64 is the Merom version of the Core 2 processor, which was released on July 27, 2006. None of Intel's earlier notebook CPUs (Core Duo, Pentium M, Celeron M, Mobile Pentium 4) implement Intel 64. Intel's processors implementing the Intel64 architecture include the Pentium 4 F-series/5x1 series, 506, and 516, Celeron D models 3x1, 3x6, 355, 347, 352, 360, and 365 and all later Celerons, all models of Xeon since "Nocona", all models of Pentium Dual-Core processors since "Merom-2M", the Atom 230, 330, D410, D425, D510, D525, N450, N455, N470, N475, N550, N570, N2600 and N2800, all versions of the Pentium D, Pentium Extreme Edition, Core 2, Core i9, Core i7, Core i5, and Core i3 processors, and the Xeon Phi 7200 series processors. X86S was a simplification of x86-64 first proposed by Intel in May 2023.[36] The new architecture would have removed support for 16-bit and 32-bit operating systems, although 32-bit programs would still run under a 64-bit OS. A compliant CPU would have no longer had legacy mode, and started directly in 64-bit long mode. There would have been a way to switch to 5-level paging without going through the unpaged mode. Specific removed features included:[37] Segmentation gates 32-bit ring 0 VT-x will no longer emulate this feature Rings 1 and 2 Ring 3 I/O port (IN/OUT) access; see port-mapped I/O String port I/O (INS/OUTS) Real mode (including huge real mode), 16-bit protected mode, VM86 16-bit addressing mode VT-x will no longer provide unrestricted mode 8259 support; the only APIC supported would be X2APIC Some unused operating system mode bits 16-bit and 32-bit Startup IPI (SIPI) The draft specification received multiple updates, reaching version 1.2 by June 2024. It was eventually abandoned as of December 2024, following the formation of the x86 Ecosystem Advisory Group by Intel and AMD.[38] Main article: x86 § APX (Advanced Performance Extensions) Advanced Performance Extensions is a 2023 Intel proposal for new instructions and an additional 16 general-purpose registers. VIA Technologies introduced their first implementation of the x86-64 architecture in 2008 after five years of development by its CPU division, Centaur Technology.[39] Codenamed "Isaiah", the 64-bit architecture was unveiled on January 24, 2008,[40] and launched on May 29 under the VIA Nano brand name.[41] The processor supports a number of VIA-specific x86 extensions designed to boost efficiency in low-power appliances. It is expected that the Isaiah architecture will be twice as fast in integer performance and four times as fast in floating-point performance as the previous-generation VIA Esther at an equivalent clock speed. Power consumption is also expected to be on par with the previous-generation VIA CPUs, with thermal design power ranging from 5 W to 25 W.[42] Being a completely new design, the Isaiah architecture was built with support for features like the x86-64 instruction set and x86 virtualization which were unavailable on its predecessors, the VIA C7 line, while retaining their encryption extensions. In 2020, through a collaboration between AMD, Intel, Red Hat, and SUSE, three microarchitecture levels (or feature levels) on top of the x86-64 baseline were defined: x86-64-v2, x86-64-v3, and x86-64-v4.[43][44] These levels define specific features that can be targeted by programmers to provide compile-time optimizations. The features exposed by each level are as follows:[45] CPU microarchitecture levels Level name CPU features Example instruction Supported processors (baseline)also as:x86-64-v1 CMOV cmov baseline for all x86-64 CPUs matches the common capabilities between the 2003 AMD AMD64 and the 2004 Intel EM64T initial implementations in the AMD K8 and the Intel Prescott processor families CX8 cmpxchg8b FPU fld FXSR fxsave MMX emms OSFXSR fxsave SCE syscall SSE cvtss2si SSE2 cvtpi2pd x86-64-v2 CMPXCHG16B cmpxchg16b Intel Nehalem and newer Intel "big" cores Intel (Atom) Silvermont and newer Intel "small" cores AMD Bulldozer and newer AMD "big" cores AMD Jaguar VIA Nano and Eden "C" features match the 2008 Intel Nehalem architecture, excluding Intel-specific instructions LAHF-SAHF lahf POPCNT popcnt SSE3 addsubpd SSE4_1 blendpd SSE4_2 pcmpestri SSSE3 pshufb x86-64-v3 AVX vzeroall Intel Haswell and newer Intel "big" cores (AVX2 enabled models only) Intel (Atom) Gracemont and newer Intel "small" cores AMD Excavator and newer AMD "big" cores QEMU emulation (as of version 7.2)[46][47] features match the 2013 Intel Haswell architecture, excluding Intel-specific instructions AVX2 vpermd BMI1 andn BMI2 bzhi F16C vcvtph2ps FMA vfmadd132pd LZCNT lzcnt MOVBE movbe OSXSAVE xgetbv x86-64-v4 AVX512F kmovw Intel Skylake and newer Intel "big" cores (AVX512 enabled models only) AMD Zen 4 and newer AMD cores features match the 2017 Intel Skylake-X architecture, excluding Intel-specific instructions AVX512BW vdbpsadbw AVX512CD vplzcntd AVX512DQ vpmullq AVX512VL — The x86-64 microarchitecture feature levels can also be found as AMD64-v1, AMD64-v2 .. or AMD64_v1 .. in settings where the "AMD64" nomenclature is used. These are used as synonyms with the x86-64-vX nomenclature and are thus functionally identical. Examples of this include the Go language documentation and the Fedora Linux distribution. All levels include features found in the previous levels. Instruction set extensions not concerned with general-purpose computation, including AES-NI and RDRAND, are excluded from the level requirements. On any x86_64 Linux distribution, all x86_64 feature levels supported by a CPU can be verified using command: ld.so --help The result will be visible at the end of command's output: Subdirectories of glibc-hwcaps directories, in priority order: x86-64-v4 x86-64-v3 (supported, searched) x86-64-v2 (supported, searched) Here x86-64-v4 feature level is not supported by CPU, but x86-64-v3 and x86-64-v2 are, which means this CPU does not support AVX512 required at v4 level. Although nearly identical, there are some differences between the two instruction sets in the semantics of a few seldom used machine instructions (or situations), which are mainly used for system programming.[48] Unless instructed to otherwise via -march settings, compilers generally produce executables (i.e. machine code) that avoid any differences, at least for ordinary application programs. This is therefore of interest mainly to developers of compilers, operating systems and similar, which must deal with individual and special system instructions. Intel 64 allows SYSCALL/SYSRET only in 64-bit mode (not in compatibility mode),[49] and allows SYSENTER/SYSEXIT in both modes.[50] AMD64 lacks SYSENTER/SYSEXIT in both sub-modes of long mode.[11]:33 When returning to a non-canonical address using SYSRET, AMD64 processors execute the general protection fault handler in privilege level 3,[51] while on Intel 64 processors it is executed in privilege level 0.[52] The SYSRET instruction will load a set of fixed values into the hidden part of the SS segment register (base-address, limit, attributes) on Intel 64 but leave the hidden part of SS unchanged on AMD64.[53] AMD64 requires a different microcode update format and control MSRs (model-specific registers), while Intel 64 implements microcode update unchanged from their 32-bit only processors. Intel 64 lacks some MSRs that are considered architectural in AMD64. These include SYSCFG, TOP_MEM, and TOP_MEM2. Intel 64 lacks the ability to save and restore a reduced (and thus faster) version of the floating-point state (involving the FXSAVE and FXRSTOR instructions).[clarification needed] In 64-bit mode, near branches with the 66H (operand size override) prefix behave differently. Intel 64 ignores this prefix: the instruction has a 32-bit sign extended offset, and instruction pointer is not truncated. AMD64 uses a 16-bit offset field in the instruction, and clears the top 48 bits of instruction pointer. On Intel 64 but not AMD64, the REX.W prefix can be used with the far-pointer instructions (LFS, LGS, LSS, JMP FAR, CALL FAR) to increase the size of their far pointer argument to 80 bits (64-bit offset + 16-bit segment). When the MOVSXD instruction is executed with a memory source operand and an operand-size of 16 bits, the memory operand will be accessed with a 16-bit read on Intel 64, but a 32-bit read on AMD64. The FCOMI/FCOMIP/FUCOMI/FUCOMIP (x87 floating-point compare) instructions will clear the OF, SF and AF bits of EFLAGS on Intel 64, but leave these flag bits unmodified on AMD64. For the VMASKMOVPS/VMASKMOVPD/VPMASKMOVD/VPMASKMOVQ (AVX/AVX2 masked move to/from memory) instructions, Intel 64 architecturally guarantees that the instructions will not cause memory faults (e.g. page-faults and segmentation-faults) for any zero-masked lanes, while AMD64 does not provide such a guarantee. If the RDRAND instruction fails to obtain a random number (as indicated by EFLAGS.CF=0), the destination register is architecturally guaranteed to be set to 0 on Intel 64 but not AMD64. For the VPINSRD and VPEXTRD (AVX vector lane insert/extract) instructions outside 64-bit mode, AMD64 requires the instructions to be encoded with VEX.W=0, while Intel 64 also accepts encodings with VEX.W=1. (In 64-bit mode, both AMD64 and Intel 64 require VEX.W=0.) The 0F 0D /r opcode with the ModR/M byte's Mod field set to 11b is a Reserved-NOP on Intel 64[54] but will cause #UD (invalid-opcode exception) on AMD64.[55] The ordering guarantees provided by some memory ordering instructions such as LFENCE and MFENCE differ between Intel 64 and AMD64: LFENCE is dispatch-serializing (enabling it to be used as a speculation fence) on Intel 64 but is not architecturally guaranteed to be dispatch-serializing on AMD64.[56] MFENCE is a fully serializing instruction (including instruction fetch serialization) on AMD64 but not Intel 64. The MOV to CR8 and INVPCID instructions are serializing on AMD64 but not Intel 64. The LMSW instruction is serializing on Intel 64 but not AMD64. WRMSR to the x2APIC ICR (Interrupt Command Register; MSR 830h) is commonly used to produce an IPI (Inter-processor interrupt) — on Intel 64[57] but not AMD64[58] CPUs, such an IPI can be reordered before an older memory store. This section needs to be updated. The reason given is: future tense relating to processors that have been out for years, dates with day and month but no year. Please help update this article to reflect recent events or newly available information. (January 2023) The AMD64 processors prior to Revision F[59] (distinguished by the switch from DDR to DDR2 memory and new sockets AM2, F and S1) of 2006 lacked the CMPXCHG16B instruction, which is an extension of the CMPXCHG8B instruction present on most post-80486 processors. Similar to CMPXCHG8B, CMPXCHG16B allows for atomic operations on octa-words (128-bit values). This is useful for parallel algorithms that use compare and swap on data larger than the size of a pointer, common in lock-free and wait-free algorithms. Without CMPXCHG16B one must use workarounds, such as a critical section or alternative lock-free approaches.[60] Its absence also prevents 64-bit Windows prior to Windows 8.1 from having a user-mode address space larger than 8 TiB.[61] The 64-bit version of Windows 8.1 requires the instruction.[62] Early AMD64 and Intel 64 CPUs lacked LAHF and SAHF instructions in 64-bit mode. AMD introduced these instructions (also in 64-bit mode) with their 90 nm (revision D) processors, starting with Athlon 64 in October 2004.[63][64] Intel introduced the instructions in October 2005 with the 0F47h and later revisions of NetBurst.[70] The 64-bit version of Windows 8.1 requires this feature.[62] Early Intel CPUs with Intel 64 also lack the NX bit of the AMD64 architecture. It was added in the stepping E0 (0F41h) Pentium 4 in October 2004.[71] This feature is required by all versions of Windows 8. Early Intel 64 implementations had a 36-bit (64 GiB) physical addressing of memory while original AMD64 implementations had a 40-bit (1 TiB) physical addressing. Intel used the 40-bit physical addressing first on Xeon MP (Potomac), launched on 29 March 2005.[72] The difference is not a difference of the user-visible ISAs. In 2007 AMD 10h-based Opteron was the first to provide a 48-bit (256 TiB) physical address space.[73][74] Intel 64's physical addressing was extended to 44 bits (16 TiB) in Nehalem-EX in

and "long" types are 32 bits wide, "long long" is 64 bits, while pointers and types derived from pointers are 64 bits wide. Kernel mode device drivers must be 64-bit versions; there is no way to run 32-bit kernel mode executables within the 64-bit operating system. User mode device drivers can be either 32-bit or 64-bit. 16-bit Windows (Win16) and DOS applications will not run on x86-64 versions of Windows due to the removal of the virtual DOS machine subsystem (NTVDM) which relied upon the ability to use virtual 8086 mode. Virtual 8086 mode cannot be entered while running in long mode. Full implementation of the NX (No Execute) page protection feature. This is also implemented on recent 32-bit versions of Windows when they are started in PAE mode. Instead of FS segment descriptor on x86 versions of the Windows NT family, GS segment descriptor is used to point to two operating system defined structures: Thread Information Block (NT_TIB) in user mode and Processor Control Region (KPCR) in kernel mode. Thus, for example, in user mode GS:0 is the address of the first member of the Thread Information Block. Maintaining this convention made the x86-64 port easier, but required AMD to retain the function of the FS and GS segments in long mode – even though segmented addressing per se is not really used by any modern operating system.[113] Early reports claimed that the operating system scheduler would not save and restore the x87 FPU machine state across thread context switches. Observed behavior shows that this is not the case: the x87 state is saved and restored, except for kernel mode-only threads (a limitation that exists in the 32-bit version as well). The most recent documentation available from Microsoft states that the x87/MMX/3DNow! instructions may be used in long mode, but that they are deprecated and may cause compatibility problems in the future.[116] (3DNow! is no longer available on AMD processors, with the exception of the PREFETCH and PREFETCHW instructions,[118] which are also supported on Intel processors as of Broadwell.) Some components like Jet Database Engine and Data Access Objects will not be ported to 64-bit architectures such as x86-64 and IA-64.[119][120][121] Microsoft Visual Studio can compile native applications to target either the x86-64 architecture, which can run only on 64-bit Microsoft Windows, or the IA-32 architecture, which can run as a 32-bit application on 32-bit Microsoft Windows or 64-bit Microsoft Windows in WoW64 emulation mode. Managed applications can be compiled either in IA-32, x86-64 or AnyCPU modes. Software created in the first two modes behave like their IA-32 or x86-64 native code counterparts respectively; When using the AnyCPU mode, however, applications in 32-bit versions of Microsoft Windows run as 32-bit applications, while they run as a 64-bit application in 64-bit editions of Microsoft Windows. The PlayStation 4 and Xbox One use AMD x86-64 processors based on the Jaguar microarchitecture.[122][123] Firmware and games are written in x86-64 code; no legacy x86 code is involved. The PlayStation 5 and Xbox Series X/S use AMD x86-64 processors based on the Zen 2 microarchitecture.[124][125] The Steam Deck uses a custom AMD x86-64 accelerated processing unit (APU) based on the Zen 2 microarchitecture.[126] Since AMD64 and Intel 64 are substantially similar, many software and hardware products use one vendor-neutral term to indicate their compatibility with both implementations. AMD's original designation for this processor architecture, "x86-64", is still used for this purpose,[2] as is the variant "x86_64".[3][4] Other companies, such as Microsoft[6] and Sun Microsystems/Oracle Corporation,[5] use the contraction "x64" in marketing material. The term IA-64 refers to the Itanium processor, and should not be confused with x86-64, as it is a completely different instruction set. Many operating systems and products, especially those that introduced x86-64 support prior to Intel's entry into the market, use the term "AMD64" or "amd64" to refer to both AMD64 and Intel 64. amd64 Most BSD systems such as FreeBSD, MidnightBSD, NetBSD and OpenBSD refer to both AMD64 and Intel 64 under the architecture name "amd64". Some Linux distributions such as Debian, Ubuntu, Gentoo Linux refer to both AMD64 and Intel 64 under the architecture name "amd64". Microsoft Windows's x64 versions use the AMD64 moniker internally to designate various components which use or are compatible with this architecture. For example, the environment variable PROCESSOR_ARCHITECTURE is assigned the value "AMD64" as opposed to "x86" in 32-bit versions, and the system directory on a Windows x64 Edition installation CD-ROM is named "AMD64", in contrast to "i386" in 32-bit versions.[127] Sun's Solaris's isalist command identifies both AMD64- and Intel 64-based systems as "amd64". Java Development Kit (JDK): the name "amd64" is used in directory names containing x86-64 files. x86_64 The Linux kernel[128] and the GNU Compiler Collection refers to 64-bit architecture as "x86_64". Some Linux distributions, such as Fedora, openSUSE, Arch Linux, Gentoo Linux refer to this 64-bit architecture as "x86_64". Apple macOS refers to 64-bit architecture as "x86-64" or "x86_64", as seen in the Terminal command arch[3] and in their developer documentation.[2][4] Breaking with most other BSD systems, DragonFly BSD refers to 64-bit architecture as "x86_64". Haiku refers to 64-bit architecture as "x86_64". x86-64/AMD64 was solely developed by AMD. Until April 2021 when the relevant patents expired, AMD held patents on techniques used in AMD64;[129][130][131] those patents had to be licensed from AMD in order to implement AMD64. Intel entered into a cross-licensing agreement with AMD, licensing to AMD their patents on existing x86 techniques, and licensing from AMD their patents on techniques used in x86-64.[132] In 2009, AMD and Intel settled several lawsuits and cross-licensing disagreements, extending their cross-licensing agreements.[133][134][135] AGESA (AMD Generic Encapsulated Software Architecture) Transient execution CPU vulnerability ^ Various names are used for the instruction set. Prior to the launch, x86-64 and x86_64 were used, while upon the release AMD named it AMD64.[1] Intel initially used the names IA-32e and EM64T before finally settling on "Intel 64" for its implementation. Some in the industry, including Apple,[2][3][4] use x86-64 and x86_64, while others, notably Sun Microsystems[5] (now Oracle Corporation) and Microsoft,[6] use x64. The BSD family of OSs and several Linux distributions[7][8] use AMD64, as does Microsoft Windows internally.[9][10] ^ In practice, 64-bit operating systems generally do not support 16-bit applications, although modern versions of Microsoft Windows contain a limited workaround that effectively supports 16-bit InstallShield and Microsoft ACME installers by silently substituting them with 32-bit code.[12] ^ The Register reported that the stepping G1 (0F49h) of Pentium 4 will sample on October 17 and ship in volume on November 14.[68] However, Intel's document says that samples are available on September 9, whereas October 17 is the "date of first availability of post-conversion material", which Intel defines as "the projected date that a customer may expect to receive the post-conversion materials. ... customers should be prepared to receive the post-converted materials on this date".[69] ^ "Debian AMD64 FAQ". Debian Wiki. Archived from the original on September 26, 2019. Retrieved May 3, 2012. ^ a b c "x86-64 Code Model". Apple. Archived from the original on June 2, 2012. Retrieved November 23, 2012. ^ a b c arch(1) – Darwin and macOS General Commands Manual ^ a b c Kevin Van Vechten (August 9, 2006). "re: Intel XNU bug report". Darwin-dev mailing list. Apple Computer. Archived from the original on February 1, 2020. Retrieved October 5, 2006. The kernel and developer tools have standardized on "x86_64" for the name of the Mach-O architecture ^ a b "Solaris 10 on AMD Opteron". Oracle. Archived from the original on July 25, 2017. Retrieved December 9, 2010. ^ a b "Microsoft 64-Bit Computing". Microsoft. Archived from the original on December 12, 2010. Retrieved December 9, 2010. ^ "AMD64 Port". Debian. Archived from the original on September 26, 2019. Retrieved November 23, 2012. ^ "Gentoo/AMD64 Project". Gentoo Project. Archived from the original on June 3, 2013. Retrieved May 27, 2013. ^ "WOW64 Implementation Details". Archived from the original on April 13, 2018. Retrieved January 24, 2016. ^ "ProcessorArchitecture Class". Archived from the original on June 3, 2017. Retrieved January 24, 2016. ^ a b c d e f g h i j k l m n o p q r s t u AMD Corporation (December 2016). "Volume 2: System Programming" (PDF). AMD64 Architecture Programmer's Manual. AMD Corporation. Archived (PDF) from the original on July 13, 2018. Retrieved March 25, 2017. ^ Raymond Chen (October 31, 2013). "If there is no 16-bit emulation layer in 64-bit Windows, how come certain 16-bit installers are allowed to run?". Archived from the original on July 14, 2021. Retrieved July 14, 2021. ^ "IBM WebSphere Application Server 64-bit Performance Demystified" (PDF). IBM Corporation. September 6, 2007. p. 14. Archived (PDF) from the original on January 25, 2022. Retrieved April 9, 2010. Figures 5, 6 and 7 also show the 32-bit version of WAS runs applications at full native hardware performance on the POWER and x86-64 platforms. Unlike some 64-bit processor architectures, the POWER and x86-64 hardware does not emulate 32-bit mode. Therefore applications that do not benefit from 64-bit features can run with full performance on the 32-bit version of WebSphere running on the above mentioned 64-bit platforms. ^ "AMD Discloses New Technologies At Microporcessor Forum" (Press release). AMD. October 5, 1999. Archived from the original on March 8, 2012. Retrieved November 9, 2010. ^ "AMD Releases x86-64 Architectural Specification; Enables Market Driven Migration to 64-Bit Computing" (Press release). AMD. August 10, 2000. Archived from the original on March 8, 2012. Retrieved November 9, 2010. ^ AMD64 Architecture Programmer's Manual (PDF). p. 1. ^ Mauerer, W. (2010). Professional Linux kernel architecture. John Wiley & Sons. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide, Part 1" (PDF). pp. 4–7. Archived (PDF) from the original on May 16, 2011. Retrieved July 10, 2019. ^ a b "BIOS and Kernel Developer's Guide (BKDG) For AMD Family 10h Processors" (PDF). p. 24. Archived (PDF) from the original on April 18, 2016. Retrieved February 27, 2016. Physical address space increased to 48 bits. ^ "Myth and facts about 64-bit Linux" (PDF). March 2, 2008. p. 7. Archived from the original (PDF) on October 10, 2010. Retrieved May 30, 2010. Physical address space increased to 48 bits ^ Shanley, Tom (1998). Pentium Pro and Pentium II System Architecture. PC System Architecture Series (Second ed.). Addison-Wesley. p. 445. ISBN 0-201-30973-4. ^ Microsoft Corporation. "What is PAE, NX, and SSE2 and why does my PC need to support them to run Windows 8 ?". Archived from the original on April 11, 2013. Retrieved March 19, 2013. ^ a b c d "Memory Limits for Windows Releases". MSDN. Microsoft. November 16, 2013. Archived from the original on January 6, 2014. Retrieved January 20, 2014. ^ "5-Level Paging and 5-Level EPT" (PDF). Intel. May 2017. Archived (PDF) from the original on December 5, 2018. Retrieved June 17, 2017. ^ US patent 9858198, Larry Seiler, "64KB page system that supports 4KB page operation", published December 29, 2016, issued January 2, 2018, assigned to Intel Corp. ^ "Opteron 6100 Series Motherboards". Supermicro Corporation. Archived from the original on June 3, 2010. Retrieved June 22, 2010. ^ "Supermicro XeonSolutions". Supermicro Corporation. Archived from the original on May 27, 2010. Retrieved June 20, 2010. ^ "Opteron 8000 Series Motherboards". Supermicro Corporation. Archived from the original on May 27, 2010. Retrieved June 20, 2010. ^ "Tyan Product Matrix". MiTEC International Corporation. Archived from the original on June 6, 2010. Retrieved June 21, 2010. ^ a b "From the AMI Archives: AMIBIOS 8 and the Transition to EFI". American Megatrends. September 8, 2017. Archived from the original on October 25, 2021. Retrieved October 25, 2021. ^ "Intel is Continuing the Yamhill Project?". Neowin. Archived from the original on June 5, 2022. Retrieved June 5, 2022. ^ "Craig Barrett confirms 64 bit address extensions for Xeon. And Prescott". The Inquirer. February 17, 2004. Archived from the original on January 12, 2013. Retrieved August 20, 2017. ^ ""A Roundup of 64-Bit Computing", from internetnews.com". Archived from the original on September 25, 2012. Retrieved September 18, 2006. ^ Lapedus, Mark (February 6, 2004). "Intel to demo 'CT' 64-bit processor line at IDF". EDN. AspenCore Media. Archived from the original on May 25, 2021. Retrieved May 25, 2021. ^ "Intel 64 Architecture". Intel. Archived from the original on June 29, 2011. Retrieved June 29, 2007. ^ "Intel Publishes "X86-S" Specification For 64-bit Only Architecture". www.phoronix.com. ^ "Envisioning a Simplified Intel Architecture for the Future". Intel. ^ Paul Alcorn (December 19, 2024). "Intel terminates x86S initiative — unilateral quest to de-bloat x86 instruction set comes to an end". Tom's Hardware. Retrieved December 20, 2024. ^ "VIA to launch new processor architecture in 1Q08" (subscription required). DigiTimes. July 25, 2007. Archived from the original on December 3, 2008. Retrieved July 25, 2007. ^ Stokes, Jon (January 23, 2008). "Isaiah revealed: VIA's new low-power architecture". Ars Technica. Archived from the original on January 27, 2008. Retrieved January 24, 2008. ^ "VIA Launches VIA Nano Processor Family" (Press release). VIA. May 29, 2008. Archived from the original on February 3, 2019. Retrieved May 25, 2017. ^ "VIA Isaiah Architecture Introduction" (PDF). VIA. January 23, 2008. Archived from the original (PDF) on September 7, 2008. Retrieved July 31, 2013. ^ Weimer, Florian (July 10, 2020). "New x86-64 micro-architecture levels". llvm-dev (Mailing list). Archived from the original on April 14, 2021. Retrieved March 11, 2021. ^ Weimer, Florian (January 5, 2021). "Building Red Hat Enterprise Linux 9 for the x86-64-v2 microarchitecture level". Red Hat developer blog. Archived from the original on February 20, 2022. Retrieved March 22, 2022. ^ "System V Application Binary Interface Low Level System Information". x86-64 psABI repo. January 29, 2021. Archived from the original on February 2, 2021. Retrieved March 11, 2021 – via GitLab. ^ "QEMU version 7.2.0 released - QEMU". www.qemu.org. Archived from the original on December 21, 2022. Retrieved January 9, 2023. ^ "ChangeLog/7.2 - QEMU". wiki.qemu.org. Archived from the original on January 9, 2023. Retrieved January 9, 2023. ^ Wasson, Scott (March 23, 2005). "64-bit computing in theory and practice". The Tech Report. Archived from the original on March 12, 2011. Retrieved March 22, 2011. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2 (2A, 2B & 2C): Instruction Set Reference, A–Z" (PDF). Intel. September 2013. pp. 4–397. Archived (PDF) from the original on October 20, 2013. Retrieved January 21, 2014. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2 (2A, 2B & 2C): Instruction Set Reference, A-Z" (PDF). Intel. September 2013. pp. 4–400. Archived (PDF) from the original on October 20, 2013. Retrieved January 21, 2014. ^ "AMD64 Architecture Programmer's Manual Volume 3: General-Purpose and System Instructions" (PDF). AMD. May 2018. p. 419. Archived (PDF) from the original on August 20, 2018. Retrieved August 2, 2018. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2 (2A, 2B & 2C): Instruction Set Reference, A-Z" (PDF). Intel. September 2014. pp. 4–412. Archived (PDF) from the original on January 13, 2015. Retrieved December 28, 2014. ^ LKML, (PATCH) x86_64, asm: Work around AMD SYSRET SS descriptor attribute issue, 23 Apr 2015. ^ Intel, Intel 64 and IA-32 Architectures Software Developer's Manual, volume 3B, order no. 253669-086US, December 2024, chapter 24.15, page 426. Archived on 11 Jan 2025. ^ AMD, AMD64 Architecture Programmer's Manual Volume 3: General-Purpose and System Instructions, order no. 24594, rev 3.36, March 2024, see description of PREFETCHW instruction on page 283. Archived on 29 Dec 2024. ^ Hadi Brais, The Significance of the x86 LFENCE Instruction, 14 May 2018. Archived on 10 Jun 2023. ^ LKML, (PATCH 5.4 55/65) x86/apic: Add extra serialization for non-serializing MSRs, 8 Feb 2021 ^ Linux kernel, git commit: x86/barrier: Do not serialize MSR accesses on AMD, 13 Nov 2023 ^ "Live Migration with AMD-V™ Extended Migration Technology" (PDF). developer.amd.com. Archived (PDF) from the original on December 6, 2022. Retrieved June 30, 2022. ^ Maged M. Michael. "Practical Lock-Free and Wait-Free LL/SC/VL Implementations Using 64-Bit CAS" (PDF). IBM. Archived (PDF) from the original on May 2, 2013. Retrieved January 21, 2014. ^ darwou (August 20, 2004). "Why is the virtual address space 4GB anyway?". The Old New Thing. Microsoft. Archived from the original on March 26, 2017. Retrieved March 25, 2017. ^ a b "System Requirements—Windows 8.1". Archived from the original on April 28, 2014. Retrieved April 27, 2014. To install a 64-bit OS on a 64-bit PC, your processor needs to support CMPXCHG16b, PrefetchW, and LAHF/SAHF. ^ Petkov, Borislav (August 10, 2009). "Re: [PATCH v2] x86: clear incorrectly forced X86_FEATURE_LAHF_LM flag". Linux kernel mailing list. Archived from the original on January 11, 2023. Retrieved June 30, 2022. ^ "Revision Guide for AMD Athlon 64 and AMD Opteron Processors" (PDF). AMD. Archived (PDF) from the original on August 24, 2009. Retrieved July 18, 2009. ^ "Product Change Notification 105224 - 01" (PDF). Intel. Archived from the original (PDF) on November 17, 2005. ^ "Intel® Pentium® D Processor 800 Sequence and Intel® Pentium® Processor Extreme Edition 840 Specification Update" (PDF). Archived (PDF) from the original on May 18, 2021. Retrieved June 30, 2022. ^ "Intel Xeon 2.8 GHz - NE80551KG0724MM / BX80551KG2800HA". CPU-World. Archived from the original on June 28, 2020. Retrieved June 30, 2022. ^ Smith, Tony (August 23, 2005). "Intel tweaks EM64T for full AMD64 compatibility". The Register. Archived from the original on June 30, 2022. Retrieved June 30, 2022. ^ "Product Change Notification 105271 – 00" (PDF). Intel. Archived from the original (PDF) on November 17, 2005. ^ 0F47h debuted in the B0 stepping of Pentium D on October 21,[65][66] but 0F48h which also supports LAHF/SAHF launched on October 10 in the dual-core Xeon.[67][a] ^ "Product Change Notification 104101 – 00" (PDF). Intel. Archived from the original (PDF) on July 16, 2004. ^ "64-bit Intel® Xeon™ Processor MP with up to 8MB L3 Cache Datasheet" (PDF). Archived (PDF) from the original on November 17, 2022. Retrieved November 17, 2022. ^ "Justin Boggs's at Microsoft PDC 2008". p. 5. Archived from the original on November 17, 2022. Retrieved November 17, 2022. ^ Waldecker, Brian. "AMD Opteron Multicore Processors" (PDF). p. 13. Archived (PDF) from the original on December 13, 2022. Retrieved November 17, 2022. ^ "Intel® Xeon® Processor 7500 Series Datasheet, Volume 2" (PDF). Archived (PDF) from the original on November 17, 2022. Retrieved November 17, 2022. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual". September 2014. p. 2-21. Archived from the original on May 14, 2019. Intel 64 architecture increases the linear address space for software to 64 bits and supports physical address space up to 46 bits. ^ Logan, Tom (November 14, 2011). "Intel Core i7-3960X Review". Archived from the original on March 28, 2016. Retrieved July 1, 2022. ^ Ye, Huaisheng. "Introduction to 5-Level Paging in 3rd Gen Intel Xeon Scalable Processors with Linux" (PDF). Lenovo. Archived (PDF) from the original on May 26, 2022. Retrieved July 1, 2022. ^ Kennedy, Patrick (November 10, 2022). "AMD EPYC Genoa Gaps Intel Xeon in Stunning Fashion". ServeTheHome. p. 2. Archived from the original on November 17, 2022. Retrieved November 17, 2022. ^ Intel, Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2, order no. 325383-086US, December 2024, see entries on BSF and BSR instructions on pages 227 and 229 (footnote 1). Archived on 1 Jan 2025. ^ "How retiring segmentation in AMD64 long mode broke VMware". Pagetable.com. November 9, 2006. Archived from the original on July 18, 2011. Retrieved May 2, 2010. ^ "VMware and CPU Virtualization Technology" (PDF). VMware. Archived (PDF) from the original on July 17, 2011. Retrieved September 8, 2010. ^ "Linux-Kernel Archive: [PATCH 2/5] KVM: svm: Disallow EFER.LMSLE on hardware that doesn't support it". lkml.indiana.edu. Retrieved November 3, 2023. ^ "Statistics | TOP500 Supercomputer Sites". Top500.org. Archived from the original on March 19, 2014. Retrieved March 22, 2014. ^ "Sublist Generator | TOP500 Supercomputer Sites". www.top500.org. Archived from the original on December 7, 2018. Retrieved December 6, 2018. ^ "Intel® Xeon PhiTM Coprocessor Instruction Set Architecture Reference Manual" (PDF). Intel. September 7, 2012. section B.2 Intel Xeon Phi coprocessor 64 bit Mode Limitations. Archived (PDF) from the original on May 21, 2014. Retrieved May 21, 2014. ^ "Intel Powers the World's Fastest Supercomputer, Reveals New and Future High Performance Computing Technologies". Archived from the original on June 22, 2013. Retrieved June 21, 2013. ^ "cvs commit: src/sys/amd64/amd64 genassym.c src/sys/amd64/include asm.h atomic.h bootinfo.h coredump.h cpufunc.h elf.h endian.h exec.h float.h fpu.h frame.h globaldata.h ieeefp.h limits.h lock.h md_var.h param.h pcb.h pcb_ext.h pmap.h proc.h profile.h psl.h ..." Archived from the original on December 4, 2008. Retrieved May 3, 2009. ^ "AMD64 port". Archived from the original on May 18, 2010. Retrieved May 3, 2009. ^ "DragonFlyBSD: GoogleSoC2008". Archived from the original on April 27, 2009. Retrieved May 3, 2009. ^ "Summer of Code accepted students". Archived from the original on September 4, 2010. Retrieved May 3, 2009. ^ "DragonFlyBSD: release24". Archived from the original on September 23, 2009. Retrieved May 3, 2009. ^ "Tutorial for entering protected and long mode from DOS". Archived from the original on February 22, 2017. Retrieved July 6, 2008. ^ Andi Kleen (June 26, 2001). "Porting Linux to x86-64". Archived from the original on September 10, 2010. Status: The kernel, compiler, tool chain work. The kernel boots and work on simulator and is used for porting of userland and running programs ^ Andi Kleen. "Andi Kleen's Page". Archived from the original on December 7, 2009. Retrieved August 21, 2009. This was the original paper describing the Linux x86-64 kernel port back when x86-64 was only available on simulators. ^ "Arch64 FAQ". April 23, 2012. Archived from the original on May 14, 2012. Retrieved May 11, 2012. You can either use the multilib packages or a i686 chroot. ^ a b Thorsten Leemhuis (September 13, 2011). "Kernel Log: x32 ABI gets around 64-bit drawbacks". www.h-online.com. Archived from the original on October 28, 2011. Retrieved November 1, 2011. ^ a b "x32 - a native 32-bit ABI for x86-64". linuxplumbersconf.org. Archived from the original on May 5, 2012. Retrieved November 1, 2011. ^ a b "x32-abi". Google Sites. Archived from the original on October 30, 2011. Retrieved November 1, 2011. ^ "AMD64 Port". debian.org. Archived from the original on September 26, 2019. Retrieved October 29, 2011. ^ "5-level paging". kernel.org. Retrieved February 7, 2024. ^ "Apple – Mac OS X Xcode 2.4 Release Notes: Compiler Tools". Apple Inc. April 11, 2007. Archived from the original on April 22, 2009. Retrieved November 19, 2012. ^ "Apple – Mac OS X Leopard – Technology - 64-bit". Apple Inc. Archived from the original on January 12, 2009. Retrieved November 19, 2012. ^ "Mac OS X v10.6: Macs that use the 64-bit kernel". Apple Inc. Archived from the original on August 31, 2009. Retrieved November 29, 2012. ^ John Siracusa. "Mac OS X 10.6 Snow Leopard: the Ars Technica review". Ars Technica LLC. Archived from the original on October 9, 2009. Retrieved June 20, 2010. ^ "Mac OS X Technology". Apple Inc. Archived from the original on March 28, 2011. Retrieved November 19, 2012. ^ Schmid, J; Thomases, K; Ramey, J; Czekalla, U; Mathieu, B; Abhiram, R (September 10, 2019). "So We Don't Have a Solution for Catalina...Yet". CodeWeavers Blog. Archived from the original on September 29, 2021. Retrieved September 29, 2021. ^ Thomases, Ken (December 11, 2019). "win32 on macOS". WineHQ. Archived from the original on November 11, 2020. Retrieved September 29, 2021. ^ "Microsoft Raises the Speed Limit with the Availability of 64-Bit Editions of Windows Server 2003 and Windows XP Professional". Microsoft News Center (Press release). April 25, 2005. Retrieved January 29, 2024. ^ "A description of the x64-based versions of Windows Server 2003 and of Windows XP Professional x64 Edition". Microsoft Support. Archived from the original on April 20, 2016. Retrieved August 14, 2016. ^ "Windows Server 2003 SP1 Administration Tools Pack". Microsoft Download Center. Archived from the original on August 27, 2016. Retrieved August 14, 2016. ^ a b "/LARGEADDRESSAWARE (Handle Large Addresses)". Visual Studio 2022 Documentation – MSVC Linker Reference – MSVC Linker Options. Microsoft. Archived from the original on December 21, 2022. Retrieved December 21, 2022. The /LARGEADDRESSAWARE option tells the linker that the application can handle addresses larger than 2 gigabytes. ^ a b c Matt Pietrek (May 2006). "Everything You Need To Know To Start Programming 64-Bit Windows Systems". Microsoft. Retrieved April 18, 2023. ^ Chris St. Amand (January 2006). "Making the Move to x64". Microsoft. Retrieved April 18, 2023. ^ "Behind Windows x86-64's 44-bit Virtual Memory Addressing Limit". Archived from the original on December 23, 2008. Retrieved July 2, 2009. ^ a b "64-bit programming for Game Developers". Retrieved April 18, 2023. ^ "Memory Limits for Windows and Windows Server Releases". Microsoft. Retrieved April 18, 2023. ^ Kingsley-Hughes, Adrian (August 23, 2010). "AMD says goodbye to 3DNow! instruction set". ZDNet. Archived from the original on January 8, 2023. Retrieved January 8, 2023. ^ "General Porting Guidelines". Programming Guide for 64-bit Windows. Microsoft Docs. Retrieved April 18, 2023. ^ "Driver history for Microsoft SQL Server". Microsoft Docs. Retrieved April 18, 2023. ^ "Microsoft OLE DB Provider for Jet and Jet ODBC driver are available in 32-bit versions only". Office Access. Microsoft Docs. KB957570. Retrieved April 18, 2023. ^ Anand Lal Shimpi (May 21, 2013). "The Xbox One: Hardware Analysis & Comparison to PlayStation 4". Anandtech. Archived from the original on June 7, 2013. Retrieved May 22, 2013. ^ "The Tech Spec Test: Xbox One Vs. PlayStation 4". Game Informer. May 21, 2013. Archived from the original on June 7, 2013. Retrieved May 22, 2013. ^ "What to expect from Sony 'PlayStation 5' launch in November". The Indian Express. August 31, 2020. Archived from the original on September 19, 2020. Retrieved September 14, 2020. ^ Cutress, Dr Ian. "Hot Chips 2020 Live Blog: Microsoft Xbox Series X System Architecture (6:00pm PT)". www.anandtech.com. Archived from the original on September 17, 2020. Retrieved September 14, 2020. ^ Hollister, Sean (November 12, 2021). "Steam Deck: Five big things we learned from Valve's developer summit". The Verge. Archived from the original on February 7, 2022. Retrieved November 12, 2021. ^ "ProcessorArchitecture Fields". Archived from the original on April 28, 2015. Retrieved September 4, 2013. ^ "An example file from Linux 3.7.8 kernel source tree displaying the usage of the term x86_64". Archived from the original on September 23, 2005. Retrieved February 17, 2013. ^ US 6877084 ^ US 6889312 ^ US 6732258 ^ "Patent Cross License Agreement Between AMD and Intel". January 1, 2001. Archived from the original on June 21, 2007. Retrieved August 23, 2009. ^ "AMD Intel Settlement Agreement". Archived from the original on July 7, 2017. Retrieved September 18, 2017. ^ Stephen Shankland and Jonathan E. Skillings (November 12, 2009). "Intel to pay AMD $1.25 billion in antitrust settlement". CNET. Archived from the original on November 8, 2012. Retrieved April 24, 2012. ^ Smith, Ryan (November 12, 2009). "AMD and Intel Settle Their Differences: AMD Gets To Go Fabless". AnandTech. Archived from the original on May 13, 2010. AMD Developer Guides, Manuals & ISA Documents x86-64: Extending the x86 architecture to 64-bits – technical talk by the architect of AMD64 (video archive), and second talk by the same speaker (video archive) AMD's "Enhanced Virus Protection" Intel tweaks EM64T for full AMD64 compatibility Analyst: Intel Reverse-Engineered AMD64 Early report of differences between Intel IA32e and AMD64 Porting to 64-bit GNU/Linux Systems, by Andreas Jaeger from GCC Summit 2003. An excellent paper explaining almost all practical aspects for a transition from 32-bit to 64-bit. Intel 64 Architecture Intel Software Network: "64 bits" TurboIRC.COM tutorials, including examples of how to of enter protected and long mode the raw way from DOS Seven Steps of Migrating a Program to a 64-bit System Memory Limits for Windows Releases Retrieved from " 2Computer architecture bit width "64-bit" redirects here. For 64-bit images in computer graphics, see Deep color. This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources: "64-bit computing" – news · newspapers · books · scholar · JSTOR (April 2023) (Learn how and when to remove this message) Computer architecture bit widths Bit 14812161824263031323645486064128256512bit slicing Application 8163264 Binary floating-point precision 16 (×½)2432 (×1)4064 (×2)80128 (×4)256 (×8) Decimal floating-point precision 3264128 vte Hex dump of the section table in a 64-bit Portable Executable File. A 64-bit word can be expressed as a sequence of 16 hexadecimal digits. In computer architecture, 64-bit integers, memory addresses, or other data units[a] are those that are 64 bits wide. Also, 64-bit central processing units (CPU) and arithmetic logic units (ALU) are those that are based on processor registers, address buses, or data buses of that size. A computer that uses such a processor is a 64-bit computer. From the software perspective, 64-bit computing means the use of machine code with 64-bit virtual memory addresses. However, not all 64-bit instruction sets support full 64-bit virtual memory addresses; x86-64 and AArch64, for example, support only 48 bits of virtual address, with the remaining 16 bits of the virtual address required to be all zeros (000...) or all ones (111...), and several 64-bit instruction sets support fewer than 64 bits of physical memory address. The term 64-bit also describes a generation of computers in which 64-bit processors are the norm. 64 bits is a word size that defines certain classes of computer architecture, buses, memory, and CPUs and, by extension, the software that runs on them. 64-bit CPUs have been used in supercomputers since the 1970s (Cray-1, 1975) and in reduced instruction set computers (RISC) based workstations and servers since the early 1990s. In 2003, 64-bit CPUs were introduced to the mainstream PC market in the form of x86-64 processors and the PowerPC G5. A 64-bit register can hold any of 264 (over 18 quintillion or 1.8×1019) different values. The range of integer values that can be stored in 64 bits depends on the integer representation used. With the two most common representations, the range is 0 through 18,446,744,073,709,551,615 (equal to 264 − 1) for representation as an (unsigned) binary number, and −9,223,372,036,854,775,808 (−263) through 9,223,372,036,854,775,807 (263 − 1) for representation as two's complement. Hence, a processor with 64-bit memory addresses can directly access 264 bytes (16 exabytes or EB) of byte-addressable memory. With no further qualification, a 64-bit computer architecture generally has integer and addressing registers that are 64 bits wide, allowing direct support for 64-bit data types and addresses. However, a CPU might have external data buses or address buses with different sizes from the registers, even larger (the 32-bit Pentium had a 64-bit data bus, for instance).[1] This section does not cite any sources. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. (April 2023) (Learn how and when to remove this message) Processor registers are typically divided into several groups: integer, floating-point, single instruction, multiple data (SIMD), control, and often special registers for address arithmetic which may have various uses and names such as address, index, or base registers. However, in modern designs, these functions are often performed by more general purpose integer registers. In most processors, only integer or address-registers can be used to address data in memory; the other types of registers cannot. The size of these registers therefore normally limits the amount of directly addressable memory, even if there are registers, such as floating-point registers, that are wider. Most high performance 32-bit and 64-bit processors (some notable exceptions are older or embedded ARM architecture (ARM) and 32-bit MIPS architecture (MIPS) CPUs) have integrated floating point hardware, which is often, but not always, based on 64-bit units of data. For example, although the x86/x87 architecture has instructions able to load and store 64-bit (and 32-bit) floating-point values in memory, the internal floating-point data and register format is 80 bits wide, while the general-purpose registers are 32 bits wide. In contrast, the 64-bit Alpha family uses a 64-bit floating-point data and register format, and 64-bit integer registers. Many computer instruction sets are designed so that a single integer register can store the memory address to any location in the computer's physical or virtual memory. Therefore, the total number of addresses to memory is often determined by the width of these registers. The IBM System/360 of the 1960s was an early 32-bit computer; it had 32-bit integer registers, although it only used the low order 24 bits of a word for addresses, resulting in a 16 MiB (16 × 10242 bytes) address space. 32-bit superminicomputers, such as the DEC VAX, became common in the 1970s, and 32-bit microprocessors, such as the Motorola 68000 family and the 32-bit members of the x86 family starting with the Intel 80386, appeared in the mid-1980s, making 32 bits something of a de facto consensus as a convenient register size. A 32-bit address register meant that 232 addresses, or 4 GB of random-access memory (RAM), could be referenced. When these architectures were devised, 4 GB of memory was so far beyond the typical amounts (4 MiB) in installations, that this was considered to be enough headroom for addressing. 4.29 billion addresses were considered an appropriate size to work with for another important reason: 4.29 billion integers are enough to assign unique references to most entities in applications like databases. Some supercomputer architectures of the 1970s and 1980s, such as the Cray-1,[2] used registers up to 64 bits wide, and supported 64-bit integer arithmetic, although they did not support 64-bit addressing. In the mid-1980s, Intel i860[3] development began culminating in a 1989 release; the i860 had 32-bit integer registers and 32-bit addressing, so it was not a fully 64-bit processor, although its graphics unit supported 64-bit integer arithmetic.[4] However, 32 bits remained the norm until the early 1990s, when the continual reductions in the cost of memory led to installations with amounts of RAM approaching 4 GB, and the use of virtual memory spaces exceeding the 4 GB ceiling became desirable for handling certain types of problems. In response, MIPS and DEC developed 64-bit microprocessor architectures, initially for high-end workstation and server machines. By the mid-1990s, HAL Computer Systems, Sun Microsystems, IBM, Silicon Graphics, and Hewlett-Packard had developed 64-bit architectures for their workstation and server systems. A notable exception to this trend were mainframes from IBM, which then used 32-bit data and 31-bit address sizes; the IBM mainframes did not include 64-bit processors until 2000. During the 1990s, several low-cost 64-bit microprocessors were used in consumer electronics and embedded applications. Notably, the Nintendo 64[5] and the PlayStation 2 had 64-bit microprocessors before their introduction in personal computers. High-end printers, network equipment, and industrial computers also used 64-bit microprocessors, such as the Quantum Effect Devices R5000.[6] 64-bit computing started to trickle down to the personal computer desktop from 2003 onward, when some models in Apple's Macintosh lines switched to PowerPC 970 processors (termed G5 by Apple), and Advanced Micro Devices (AMD) released its first 64-bit x86-64 processor. Physical memory eventually caught up with 32-bit limits. In 2023, laptop computers were commonly equipped with 16GB and servers starting from 64 GB of memory,[7] greatly exceeding the 4 GB address capacity of 32 bits. 1961 IBM delivers the IBM 7030 Stretch supercomputer, which uses 64-bit data words and 32- or 64-bit instruction words. 1974 Control Data Corporation launches the CDC Star-100 vector supercomputer, which uses a 64-bit word architecture (prior CDC systems were based on a 60-bit architecture). International Computers Limited launches the ICL 2900 Series with 32-bit, 64-bit, and 128-bit two's complement integers; 64-bit and 128-bit floating point; 32-bit, 64-bit, and 128-bit packed decimal and a 128-bit accumulator register. The architecture has survived through a succession of ICL and Fujitsu machines. The latest is the Fujitsu Supernova, which emulates the original environment on 64-bit Intel processors. 1976 Cray Research delivers the first Cray-1 supercomputer, which is based on a 64-bit word architecture and will form the basis for later Cray vector supercomputers. 1983 Elxsi launches the Elxsi 6400 parallel minisupercomputer. The Elxsi architecture has 64-bit data registers but a 32-bit address space. 1989 Intel introduces the Intel i860 reduced instruction set computer (RISC) processor. Marketed as a "64-Bit Microprocessor", it had essentially a 32-bit architecture, enhanced with a 3D graphics unit capable of 64-bit integer operations.[8] 1993 Atari introduces the Atari Jaguar video game console, which includes some 64-bit wide data paths in its architecture.[9] 1991 MIPS Computer Systems produces the first 64-bit microprocessor, the R4000, which implements the MIPS III architecture, the third revision of its MIPS architecture.[10] The CPU is used in SGI graphics workstations starting with the IRIS Crimson. Kendall Square Research deliver their first KSR1 supercomputer, based on a proprietary 64-bit RISC processor architecture running OSF/1. 1992 Digital Equipment Corporation (DEC) introduces the pure 64-bit Alpha architecture which was born from the PRISM project.[11] 1994 Intel announces plans for the 64-bit IA-64 architecture (jointly developed with Hewlett-Packard) as a successor to its 32-bit IA-32 processors. A 1998 to 1999 launch date was targeted. 1995 Sun launches a 64-bit SPARC processor, the UltraSPARC.[12] Fujitsu-owned HAL Computer Systems launches workstations based on a 64-bit CPU, HAL's independently designed first-generation SPARC64. IBM releases the A10 and A30 microprocessors, the first 64-bit PowerPC AS processors.[13] IBM also releases a 64-bit AS/400 system upgrade, which can convert the operating system, database and applications. 1996 Nintendo introduces the Nintendo 64 video game console, built around a low-cost variant of the MIPS R4000. HP releases the first implementation of its 64-bit PA-RISC 2.0 architecture, the PA-8000.[14] 1998 IBM releases the POWER3 line of full-64-bit PowerPC/POWER processors.[15] 1999 Intel releases the instruction set for the IA-64 architecture. AMD publicly discloses its set of 64-bit extensions to IA-32, called x86-64 (later branded AMD64). 2000 IBM ships its first 64-bit z/Architecture mainframe, the zSeries z900. z/Architecture is a 64-bit version of the 32-bit ESA/390 architecture, a descendant of the 32-bit System/360 architecture. 2001 Intel ships its IA-64 processor line, after repeated delays in getting to market. Now branded Itanium and targeting high-end servers, sales fail to meet expectations. 2003 AMD introduces its Opteron and Athlon 64 processor lines, based on its AMD64 architecture which is the first x86-based 64-bit processor architecture. Apple also ships the 64-bit "G5" PowerPC 970 CPU produced by IBM. Intel maintains that its Itanium chips would remain its only 64-bit processors. 2004 Intel, reacting to the market success of AMD, admits it has been developing a clone of the AMD64 extensions named IA-32e (later renamed EM64T, then yet again renamed to Intel 64). Intel ships updated versions of its Xeon and Pentium 4 processor families supporting the new 64-bit instruction set. VIA Technologies announces the Isaiah 64-bit processor.[16] 2006 Sony, IBM, and Toshiba begin manufacturing the 64-bit Cell processor for use in the PlayStation 3, servers, workstations, and other appliances. Intel released Core 2 Duo as the first mainstream x86-64 processor for its mobile, desktop, and workstation line. Prior 64-bit extension processor lines were not widely available in the consumer retail market (most of 64-bit Pentium 4/D were OEM), 64-bit Pentium 4, Pentium D, and Celeron were not into mass production until late 2006 due to poor yield issue (most of good yield wafers were targeted at server and mainframe while mainstream still remain 130 nm 32-bit processor line until 2006) and soon became low end after Core 2 debuted. AMD released their first 64-bit mobile processor and manufactured in 90 nm. 2011 ARM Holdings announces ARMv8-A, the first 64-bit version of the ARM architecture family.[17] 2012 ARM Holdings announced their Cortex-A53 and Cortex-A57 cores, their first cores based on their 64-bit architecture, on 30 October 2012.[18][19] 2013 Apple announces the iPhone 5S, with the world's first 64-bit processor in a smartphone, which uses their A7 ARMv8-A-based system-on-a-chip alongside the iPad Air and iPad Mini 2 which are the world's first 64-bit processor in a tablet. 2014 RISC-V was published[20]. Google announces the Nexus 9 tablet, the first Android device to run on the 64-bit Tegra K1 chip. 2015 Apple announces the iPod Touch (6th generation), the first iPod Touch to use the 64-bit processor A8 ARMv8-A-based system-on-a-chip alongside the Apple TV (4th generation) which is the world's first 64-bit processor in an Apple TV. 2018 Apple announces the Apple Watch Series 4, the first Apple Watch to use the 64-bit processor S4 ARMv8-A-based system-on-a-chip. 2020 Synopsis announce the ARCv3 ISA, the first 64-bit version of the ARC ISA.[21] 1985 Cray releases UNICOS, the first 64-bit implementation of the Unix operating system.[22] 1993 DEC releases the 64-bit DEC OSF/1 AXP Unix-like operating system (later renamed Tru64 UNIX) for its systems based on the Alpha architecture. 1994 Support for the R8000 processor is added by Silicon Graphics to the IRIX operating system in release 6.0. 1995 DEC releases OpenVMS 7.0, the first full 64-bit version of OpenVMS for Alpha. First 64-bit Linux distribution for the Alpha architecture is released.[23] 1996 Support for the R4x00 processors in 64-bit mode is added by Silicon Graphics to the IRIX operating system in release 6.2. 1998 Sun releases Solaris 7, with full 64-bit UltraSPARC support. 2000 IBM releases z/OS, a 64-bit operating system descended from MVS, for the new zSeries 64-bit mainframes; 64-bit Linux on z Systems follows the CPU release almost immediately. 2001 Linux becomes the first OS kernel to fully support x86-64 (on a simulator, as no x86-64 processors had been released yet).[24] 2001 Microsoft releases Windows XP 64-Bit Edition for the Itanium's IA-64 architecture; it could run 32-bit applications through an execution layer.[citation needed] 2003 Apple releases its Mac OS X 10.3 "Panther" operating system which adds support for native 64-bit integer arithmetic on PowerPC 970 processors.[25] Several Linux distributions release with support for AMD64. FreeBSD releases with support for AMD64. 2005 On January 4, Microsoft discontinues Windows XP 64-Bit Edition, as no PCs with IA-64 processors had been available since the previous September, and announces that it is developing x86-64 versions of Windows to replace it.[26] On January 31, Sun releases Solaris 10 with support for AMD64 and EM64T processors. On April 29, Apple releases Mac OS X 10.4 "Tiger" which provides limited support for 64-bit command-line applications on machines with PowerPC 970 processors; later versions for Intel-based Macs supported 64-bit command-line applications on Macs with EM64T processors. On April 30, Microsoft releases Windows XP Professional x64 Edition and Windows Server 2003 x64 Edition for AMD64 and EM64T processors.[27] 2006 Microsoft releases Windows Vista, including a 64-bit version for AMD64/EM64T processors that retains 32-bit compatibility. In the 64-bit version, all Windows applications and components are 64-bit, although many also have their 32-bit versions included for compatibility with plug-ins.[citation needed] 2007 Apple releases Mac OS X 10.5 "Leopard", which fully supports 64-bit applications on machines with PowerPC 970 or EM64T processors.[citation needed] 2009 Microsoft releases Windows 7, which, like Windows Vista, includes a full 64-bit version for AMD64/Intel 64 processors; most new computers are loaded by default with a 64-bit version. Microsoft also releases Windows Server 2008 R2, which is the first 64-bit only server operating system. Apple releases Mac OS X 10.6, "Snow Leopard", which ships with a 64-bit kernel for AMD64/Intel64 processors, although only certain recent models of Apple computers will run the 64-bit kernel by default. Most applications bundled with Mac OS X 10.6 are now also 64-bit.[25] 2011 Apple releases Mac OS X 10.7, "Lion", which runs the 64-bit kernel by default on supported machines. Older machines that are unable to run the 64-bit kernel run the 32-bit kernel, but, as with earlier releases, can still run 64-bit applications; Lion does not support machines with 32-bit processors. Nearly all applications bundled with Mac OS X 10.7 are now also 64-bit, including iTunes.[citation needed] 2012 Microsoft releases Windows 8 which supports UEFI Class 3 (UEFI without CSM) and Secure Boot.[28] Apple releases OS X Mountain Lion, which makes the 64-bit kernel the default on some older previously unsupported machines and removes the 32-bit kernel. 2013 Apple releases iOS 7, which, on machines with AArch64 processors, has a 64-bit kernel that supports 64-bit applications.[citation needed] 2014 Google releases Android Lollipop, the first version of the Android operating system with support for 64-bit processors. [citation needed] 2017 Apple releases iOS 11, supporting only machines with AArch64 processors. It has a 64-bit kernel that only supports 64-bit applications. 32-bit applications are no longer compatible.[citation needed] 2018 Apple releases watchOS 5, the first watchOS version to bring the 64-bit support.[citation needed] 2019 Apple releases macOS 10.15 "Catalina", dropping support for 32-bit Intel applications.[citation needed] 2021 Microsoft releases Windows 11 on October 5, which only supports 64-bit systems, dropping support for IA-32 and AArch32 systems.[citation needed] 2022 Google releases the Pixel 7, which drops support for 32-bit applications. Apple releases watchOS 9, the first watchOS version to run exclusively on the Apple Watch models with 64-bit processors (including Apple Watch Series 4 or newer, Apple Watch SE (1st generation) or newer and the newly introduced Apple Watch Ultra), dropping support for Apple Watch Series 3 as the final Apple Watch model with 32-bit processor.[citation needed] 2024 Microsoft releases Windows 11 2024 Update, ARM versions of which drop support for 32-bit ARM applications. In principle, a 64-bit microprocessor can address 16 EB (16 × 10246 = 264 = 18,446,744,073,709,551,616 bytes) of memory. However, not all instruction sets, and not all processors implementing those instruction sets, support a full 64-bit virtual or physical address space. The x86-64 architecture (as of March 2024[update]) allows 48 bits for virtual memory and, for any given processor, up to 52 bits for physical memory.[29][30] These limits allow memory sizes of 256 TB (256 × 10244 bytes) and 4 PB (4 × 10245 bytes), respectively. A PC cannot currently contain 4 petabytes of memory (due to the physical size of the memory chips), but AMD envisioned large servers, shared memory clusters, and other uses of physical address space that might approach this in the foreseeable future. Thus the 52-bit physical address provides ample room for expansion while not incurring the cost of implementing full 64-bit physical addresses. Similarly, the 48-bit virtual address space was designed to provide 65,536 (216) times the 32-bit limit of 4 GB (4 × 10243 bytes), allowing room for later expansion and incurring no overhead of translating full 64-bit addresses. The Power ISA v3.0 allows 64 bits for an effective address, mapped to a segmented address with between 65 and 78 bits allowed, for virtual memory, and, for any given processor, up to 60 bits for physical memory.[31] The Oracle SPARC Architecture 2015 allows 64 bits for virtual memory and, for any given processor, between 40 and 56 bits for physical memory.[32] The ARM AArch64 Virtual Memory System Architecture allows from 48 to 56 bits for virtual memory and, for any given processor, from 32 to 56 bits for physical memory.[33] The DEC Alpha specification requires minimum of 43 bits of virtual memory address space (8 TB) to be supported, and hardware need to check and trap if the remaining unsupported bits are zero (to support compatibility on future processors). Alpha 21064 supported 43 bits of virtual memory address space (8 TB) and 34 bits of physical memory address space (16 GB). Alpha 21164 supported 43 bits of virtual memory address space (8 TB) and 40 bits of physical memory address space (1 TB). Alpha 21264 supported user-configurable 43 or 48 bits of virtual memory address space (8 TB or 256 TB) and 44 bits of physical memory address space (16 TB). A change from a 32-bit to a 64-bit architecture is a fundamental alteration, as most operating systems must be extensively modified to take advantage of the new architecture, because that software has to manage the actual memory addressing hardware.[34] Other software must also be ported to use the new abilities; older 32-bit software may be supported either by virtue of the 64-bit instruction set being a superset of the 32-bit instruction set, so that processors that support the 64-bit instruction set can also run code for the 32-bit instruction set, or through software emulation, or by the actual implementation of a 32-bit processor core within the 64-bit processor, as with some Itanium processors from Intel, which included an IA-32 processor core to run 32-bit x86 applications. The operating systems for those 64-bit architectures generally support both 32-bit and 64-bit applications.[35] One significant exception to this is the IBM AS/400, software for which is compiled into a virtual instruction set architecture (ISA) called Technology Independent Machine Interface (TIMI); TIMI code is then translated to native machine code by low-level software before being executed. The translation software is all that must be rewritten to move the full OS and all software to a new platform, as when IBM transitioned the native instruction set for AS/400 from the older 32/48-bit IMPI to the newer 64-bit PowerPC-AS, codenamed Amazon. The IMPI instruction set was quite different from even 32-bit PowerPC, so this transition was even bigger than moving a given instruction set from 32 to 64 bits. On 64-bit hardware with x86-64 architecture (AMD64), most 32-bit operating systems and applications can run with no compatibility issues. While the larger address space of 64-bit architectures makes working with large data sets in applications such as digital video, scientific computing, and large databases easier, there has been considerable debate on whether they or their 32-bit compatibility modes will be faster than comparably priced 32-bit systems for other tasks. A compiled Java program can run on a 32- or 64-bit Java virtual machine with no modification. The lengths and precision of all the built-in types, such as char, short, int, long, float, and double, and the types that can be used as array indices, are specified by the standard and are not dependent on the underlying architecture. Java programs that run on a 64-bit Java virtual machine have access to a larger address space.[36] Speed is not the only factor to consider in comparing 32-bit and 64-bit processors. Applications such as multi-tasking, stress testing, and clustering – for high-performance computing (HPC) – may be more suited to a 64-bit architecture when deployed appropriately. For this reason, 64-bit clusters have been widely deployed in large organizations, such as IBM, HP, and Microsoft. Summary: A 64-bit processor performs best with 64-bit software. A 64-bit processor may have backward compatibility, allowing it to run 32-bit application software for the 32-bit version of its instruction set, and may also support running 32-bit operating systems for the 32-bit version of its instruction set. A 32-bit processor is incompatible with 64-bit software. A common misconception is that 64-bit architectures are no better than 32-bit architectures unless the computer has more than 4 GB of random-access memory.[37] This is not entirely true: Some operating systems and certain hardware configurations limit the physical memory space to 3 GB on IA-32 systems, due to much of the 3–4 GB region being reserved for hardware addressing; see 3 GB barrier; 64-bit architectures can address far more than 4 GB. However, IA-32 processors from the Pentium Pro onward allow a 36-bit physical memory address space, using Physical Address Extension (PAE), which gives a 64 GB physical address range, of which up to 62 GB may be used by main memory; operating systems that support PAE may not be limited to 4 GB of physical memory, even on IA-32 processors. However, drivers and other kernel mode software, more so older versions, may be incompatible with PAE; this has been cited as the reason for 32-bit versions of Microsoft Windows being limited to 4 GB of physical RAM[38] (although the validity of this explanation has been disputed[39]). Some operating systems reserve portions of process address space for OS use, effectively reducing the total address space available for mapping memory for user programs. For instance, 32-bit Windows reserves 1 or 2 GB (depending on the settings) of the total address space for the kernel, which leaves only 3 or 2 GB (respectively) of the address space available for user mode. This limit is much higher on 64-bit operating systems. Memory-mapped files are becoming more difficult to implement in 32-bit architectures as files of over 4 GB become more common; such large files cannot be memory-mapped easily to 32-bit architectures, as only part of the file can be mapped into the address space at a time, and to access such a file by memory mapping, the parts mapped must be swapped into and out of the address space as needed. This is a problem, as memory mapping, if properly implemented by the OS, is one of the most efficient disk-to-memory methods. Some 64-bit programs, such as encoders, decoders and encryption software, can benefit greatly from 64-bit registers,[citation needed] while the performance of other programs, such as 3D graphics-oriented ones, remains unaffected when switching from a 32-bit to a 64-bit environment.[citation needed] Some 64-bit architectures, such as x86-64 and AArch64, support more general-purpose registers than their 32-bit counterparts (although this is not due specifically to the word length). This leads to a significant speed increase for tight loops since the processor does not have to fetch data from the cache or main memory if the data can fit in the available registers. Example in C: int a, b, c, d, e; for (a = 0; a < 100; a++) { b = a; c = b; d = c; e = d; } This code first creates 5 values: a, b, c, d and e; and then puts them in a loop. During the loop, this code changes the value of b to the value of a, the value of c to the value of b, the value of d to the value of c and the value of e to the value of d. This has the same effect as changing all the values to a. If a processor can keep only two or three values or variables in registers, it would need to move some values between memory and registers to be able to process variables d and e also; this is a process that takes many CPU cycles. A processor that can hold all values and variables in registers can loop through them with no need to move data between registers and memory for each iteration. This behavior can easily be compared with virtual memory, although any effects are contingent on the compiler. The main disadvantage of 64-bit architectures is that, relative to 32-bit architectures, the same data occupies more space in memory (due to longer pointers and possibly other types, and alignment padding). This increases the memory requirements of a given process and can have implications for efficient processor cache use. Maintaining a partial 32-bit model is one way to handle this, and is in general reasonably effective. For example, the z/OS operating system takes this approach, requiring program code to reside in 31-bit address spaces (the high order bit is not used in address calculation on the underlying hardware platform) while data objects can optionally reside in 64-bit regions. Not all such applications require a large address space or manipulate 64-bit data items, so these applications do not benefit from these features. x86-based 64-bit systems sometimes lack equivalents of software that is written for 32-bit architectures. The most severe problem in Microsoft Windows is incompatible device drivers for obsolete hardware. Most 32-bit application software can run on a 64-bit operating system in a compatibility mode, also termed an emulation mode, e.g., Microsoft WoW64 Technology for IA-64 and AMD64. The 64-bit Windows Native Mode[40] driver environment runs atop 64-bit NTDLL.DLL, which cannot call 32-bit Win32 subsystem code (often devices whose actual hardware function is emulated in user mode software, like Winprinters). Because 64-bit drivers for most devices were unavailable until early 2007 (Vista x64), using a 64-bit version of Windows was considered a challenge. However, the trend has since moved toward 64-bit computing, more so as memory prices dropped and the use of more than 4 GB of RAM increased. Most manufacturers started to provide both 32-bit and 64-bit drivers for new devices, so unavailability of 64-bit drivers ceased to be a problem. 64-bit drivers were not provided for many older devices, which could consequently not be used in 64-bit systems. Driver compatibility was less of a problem with open-source drivers, as 32-bit ones could be modified for 64-bit use. Support for hardware made before early 2007, was problematic for open-source platforms,[citation needed] due to the relatively small number of users. 64-bit versions of Windows cannot run 16-bit software. However, most 32-bit applications will work well. 64-bit users are forced to install a virtual machine of a 16- or 32-bit operating system to run 16-bit applications or use one of the alternatives for NTVDM.[41] Mac OS X 10.4 "Tiger" and Mac OS X 10.5 "Leopard" had only a 32-bit kernel, but they can run 64-bit user-mode code on 64-bit processors. Mac OS X 10.6 "Snow Leopard" had both 32- and 64-bit kernels, and, on most Macs, used the 32-bit kernel even on 64-bit processors. This allowed those Macs to support 64-bit processes while still supporting 32-bit device drivers; although not 64-bit drivers and performance advantages that can come with them. Mac OS X 10.7 "Lion" ran with a 64-bit kernel on more Macs, and OS X 10.8 "Mountain Lion" and later macOS releases only have a 64-bit kernel. On systems with 64-bit processors, both the 32- and 64-bit macOS kernels can run 32-bit user-mode code, and all versions of macOS up to macOS Mojave (10.14) include 32-bit versions of libraries that 32-bit applications would use, so 32-bit user-mode software for macOS will run on those systems. The 32-bit versions of libraries have been removed by Apple in macOS Catalina (10.15). Linux and most other Unix-like operating systems, and the C and C++ toolchains for them, have supported 64-bit processors for many years. Many applications and libraries for those platforms are open-source software, written in C and C++, so that if they are 64-bit-safe, they can be compiled into 64-bit versions. This source-based distribution model, with an emphasis on frequent releases, makes availability of application software for those operating systems less of an issue. In 32-bit programs, pointers and data types such as integers generally have the same length. This is not necessarily true on 64-bit machines.[42][43][44] Mixing data types in programming languages such as C and its descendants such as C++ and Objective-C may thus work on 32-bit implementations but not on 64-bit implementations. In many programming environments for C and C-derived languages on 64-bit machines, int variables are still 32 bits wide, but long integers and pointers are 64 bits wide. These are described as having an LP64 data model, which is an abbreviation of "Long, Pointer, 64".[45][46] Other models are the ILP64 data model in which all three data types are 64 bits wide,[47][46] and even the SILP64 model where short integers are also 64 bits wide.[48][49] However, in most cases the modifications required are relatively minor and straightforward, and many well-written programs can simply be recompiled for the new environment with no changes. Another alternative is the LLP64 model, which maintains compatibility with 32-bit code by leaving both int and long as 32-bit.[50][46] LL refers to the long long integer type, which is at least 64 bits on all platforms, including 32-bit environments. There are also systems with 64-bit processors using an ILP32 data model, with the addition of 64-bit long long integers; this is also used on many platforms with 32-bit processors. This model reduces code size and the size of data structures containing pointers, at the cost of a much smaller address space, a good choice for some embedded systems. For instruction sets such as x86 and ARM in which the 64-bit version of the instruction set has more registers than does the 32-bit version, it provides access to the additional registers without the space penalty. It is common in 64-bit RISC machines,[citation needed] explored in x86 as x32 ABI, and has recently been used in the Apple Watch Series 4 and 5.[51][52] 64-bit data models Datamodel shortint int longint longlong Pointer,size_t Sample operating systems ILP32 16 32 32 64 32 x32 and arm64ilp32 ABIs on Linux systems; MIPS N32 ABI. LLP64 16 32 32 64 64 Microsoft Windows (x86-64, IA-64, and ARM64) using Visual C++; and MinGW LP64 16 32 64 64 64 Most Unix and Unix-like systems, e.g., Solaris, Linux, BSD, macOS. Windows when using Cygwin; z/OS ILP64 16 64 64 64 64 HAL Computer Systems port of Solaris to the SPARC64 SILP64 64 64 64 64 64 Classic UNICOS[48][49] (versus UNICOS/mp, etc.) Many 64-bit platforms today use an LP64 model (including Solaris, AIX, HP-UX, Linux, macOS, BSD, and IBM z/OS). Microsoft Windows uses an LLP64 model. The disadvantage of the LP64 model is that storing a long into an int truncates. On the other hand, converting a pointer to a long will "work" in LP64. In the LLP64 model, the reverse is true. These are not problems which affect fully standard-compliant code, but code is often written with implicit assumptions about the widths of data types. C code should prefer (u)intptr_t instead of long when casting pointers into integer objects. A programming model is a choice made to suit a given compiler, and several can coexist on the same OS. However, the programming model chosen as the primary model for the OS application programming interface (API) typically dominates. Another consideration is the data model used for device drivers. Drivers make up the majority of the operating system code in most modern operating systems[citation needed] (although many may not be loaded when the operating system is running). Many drivers use pointers heavily to manipulate data, and in some cases have to load pointers of a certain size into the hardware they support for direct memory access (DMA). As an example, a driver for a 32-bit PCI device asking the device to DMA data into upper areas of a 64-bit machine's memory could not satisfy requests from the operating system to load data from the device to memory above the 4 gigabyte barrier, because the pointers for those addresses would not fit into the DMA registers of the device. This problem is solved by having the OS take the memory restrictions of the device into account when generating requests to drivers for DMA, or by using an input–output memory management unit (IOMMU). This section does not cite any sources. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. (April 2023) (Learn how and when to remove this message) As of August 2023[update], 64-bit architectures for which processors are being manufactured include: The 64-bit extension created by Advanced Micro Devices (AMD) to Intel's x86 architecture (later licensed by Intel); commonly termed x86-64, AMD64, or x64: AMD's AMD64 extensions (used in Athlon 64, Opteron, Sempron, Turion 64, Phenom, Athlon II, Phenom II, APU, FX, Ryzen, and Epyc processors) Intel's Intel 64 extensions, used in Intel Core 2/i3/i5/i7/i9, some Atom, and newer Celeron, Pentium, and Xeon processors Intel's K1OM architecture, a variant of Intel 64 with no CMOV, MMX, and SSE instructions, used in first-generation Xeon Phi (Knights Corner) coprocessors, binary incompatible with x86-64 programs VIA Technologies' 64-bit extensions, used in the VIA Nano processors ARM Holdings' AArch64 architecture IBM's PowerPC/Power ISA: IBM's Power10 processor and predecessors, and the IBM A2 processors IBM's z/Architecture, a 64-bit version of the ESA/390 architecture, used in IBM's IBM Z mainframes: IBM Telum II processor and predecessors Hitachi AP8000E RISC-V SPARC V9 architecture: Oracle's M8 and S7 processors and predecessors Fujitsu's SPARC64 XII and SPARC64 XIfx processors and predecessors MIPS Technologies' MIPS64 architecture NEC SX architecture SX-Aurora TSUBASA Elbrus architecture: Elbrus-8S ARC Most architectures of 64 bits that are derived from the same architecture of 32 bits can execute code written for the 32-bit versions natively, with no performance penalty.[citation needed] This kind of support is commonly called bi-arch support or more generally multi-arch support. Computer memory ^ such as floating-point numbers. ^ Pentium Processor User's Manual Volume 1: Pentium Processor Data Book (PDF). Intel. 1993. ^ "Cray-1 Computer System Hardware Reference Manual" (PDF). Cray Research. 1977. Retrieved October 8, 2013. ^ Grimes, Jack; Kohn, Les; Bharadhwaj, Rajeev (July–August 1989). "The Intel i860 64-Bit Processor: A General-Purpose CPU with 3D Graphics Capabilities". IEEE Computer Graphics and Applications. 9 (4): 85–94. doi:10.1109/38.31467. S2CID 38831149. Retrieved 2010-11-19. ^ "i860 Processor Family Programmer's Reference Manual" (PDF). Intel. 1991. Retrieved September 12, 2019. ^ "NEC Offers Two High Cost Performance 64-bit RISC Microprocessors" (Press release). NEC. 1998-01-20. Retrieved 2011-01-09. Versions of the VR4300 processor are widely used in consumer and office automation applications, including the popular Nintendo 64™ video game and advanced laser printers such as the recently announced, award-winning Hewlett-Packard LaserJet 4000 printer family. ^ MIPS R5000 Microprocessor Technical Backgrounder (PDF), MIPS Technologies, Inc, retrieved 2024-08-19 ^ "DDR5 | DRAM". Samsung Semiconductor Global. Retrieved 2025-01-19. ^ "i860 64-Bit Microprocessor". Intel. 1989. Archived from the original on 19 March 2011. Retrieved 30 November 2010. ^ "Atari Jaguar History". AtariAge. ^ Joe Heinrich (1994). MIPS R4000 Microprocessor User's Manual (2nd ed.). MIPS Technologies, Inc. ^ Richard L. Sites (1992). "Alpha AXP Architecture". Digital Technical Journal. 4 (4). Digital Equipment Corporation. ^ Gwennap, Linley (3 October 1994). "UltraSparc Unleashes SPARC Performance". Microprocessor Report. 8 (13). MicroDesign Resources. ^ Bishop, J. W.; et al. (July 1996). "PowerPC AS A10 64-bit RISC microprocessor". IBM Journal of Research and Development. 40 (4). IBM Corporation: 495–505. doi:10.1147/rd.404.0495. ^ Gwennap, Linley (14 November 1994). "PA-8000 Combines Complexity and Speed". Microprocessor Report. 8 (15). MicroDesign Resources. ^ F. P. O'Connell; S. W. White (November 2000). "POWER3: The next generation of PowerPC processors". IBM Journal of Research and Development. 44 (6). IBM Corporation: 873–884. doi:10.1147/rd.446.0873. ^ "VIA Unveils Details of Next-Generation Isaiah Processor Core" (Press release). VIA Technologies, Inc. Archived from the original on 2007-10-11. Retrieved 2007-07-18. ^ "ARMv8 Technology Preview" (PDF). October 31, 2011. Archived from the original (PDF) on November 11, 2011. Retrieved November 15, 2012. ^ "ARM Launches Cortex-A50 Series, the World's Most Energy-Efficient 64-bit Processors" (Press release). ARM Holdings. Retrieved 2012-10-31. ^ "ARM Keynote: ARM Cortex-A53 and ARM Cortex-A57 64bit ARMv8 processors launched". ARMdevices.net. 2012-10-31. ^ Asanović, Krste; Patterson, David A. (August 6, 2014). Instruction Sets Should Be Free: The Case For RISC-V (PDF). EECS Department, University of California, Berkeley. UCB/EECS-2014-146. ^ "Synopsys Introduces New 64-bit ARC Processor IP". Archived from the original on 31 March 2022. ^ Stefan Berka. "Unicos Operating System". www.operating-system.org. Archived from the original on 26 November 2010. Retrieved 2010-11-19. ^ Jon "maddog" Hall (Jun 1, 2000). "My Life and Free Software". Linux Journal. ^ Andi Kleen. Porting Linux to x86-64 (PDF). Ottawa Linux Symposium 2001. Status: The kernel, compiler, tool chain work. The kernel boots and work on simulator and is used for porting of userland and running programs ^ a b John Siracusa (September 2009). "Mac OS X 10.6 Snow Leopard: the Ars Technica review". Ars Technica. p. 5. Archived from the original on 9 October 2009. Retrieved 2009-09-06. ^ Joris Evers (5 January 2005). "Microsoft nixes Windows XP for Itanium". Computerworld. Archived from the original on 18 June 2013. Retrieved 17 October 2017. ^ "Microsoft Raises the Speed Limit with the Availability of 64-Bit Editions of Windows Server 2003 and Windows XP Professional" (Press release). Microsoft. April 25, 2005. Retrieved September 10, 2015. ^ "UEFI on Dell BizClient Platforms" (PDF). ^ "AMD64 Programmer's Manual Volume 2: System Programming" (PDF). Advanced Micro Devices. March 2024. p. 127. ^ "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1" (PDF). Intel. September 2016. p. 4-2. ^ "Power ISA Version 3.0". IBM. November 30, 2015. p. 983. ^ "Oracle SPARC Architecture 2015 Draft D1.0.9" (PDF). Oracle. November 16, 2016. p. 475. Archived from the original (PDF) on April 22, 2020. ^ "ARM Architecture Reference Manual for A-profile architecture". 30 November 2024. section D8.1.6 "Implemented physical address size", section D8.1.8 "Supported virtual address ranges". ^ Mashey, John (October 2006). "The Long Road to 64 Bits". ACM Queue. 4 (8): 85–94. doi:10.1145/1165754.1165766. ^ "Windows 7: 64 bit vs 32 bit?". W7 Forums. 2 April 2009. Archived from the original on 5 April 2009. Retrieved 2009-04-05. ^ "Frequently Asked Questions About the Java HotSpot VM". Oracle. Retrieved 2024-12-13. ^ "A description of the differences between 32-bit versions of Windows Vista and 64-bit versions of Windows Vista". Archived from the original on 2011-10-15. Retrieved 2011-10-14. ^ Mark Russinovich (2008-07-21). "Pushing the Limits of Windows: Physical Memory". Archived from the original on

2017-04-07. Retrieved 2017-03-09. ^ Chappell, Geoff (2009-01-27). "Licensed Memory in 32-Bit Windows Vista". geoffchappell.com. WP:SPS. Retrieved 9 March 2017. ^ "Inside Native Applications". Technet.microsoft.com. 2006-11-01. Archived from the original on 23 October 2010. Retrieved 2010-11-19. ^ Lincoln Spector (August 12, 2013). "Run an old program on a new PC". ^ Peter Seebach (2006). "Exploring 64-bit development on POWER5: How portable is your code, really?". IBM. ^ Henry Spencer. "The Ten Commandments for C Programmers". ^ "The Story of Thud and Blunder". Datacenterworks.com. Retrieved 2010-11-19. ^ "ILP32 and LP64 data models and data type sizes". z/OS XL C/C++ Programming Guide. ^ a b c "64-Bit Programming Models". Retrieved 2020-06-05. ^ "Using the ILP64 Interface vs. LP64 Interface". Intel. Retrieved Jun 24, 2020. ^ a b "Cray C/C++ Reference Manual". August 1998. Table 9-1. Cray Research systems data type mapping. Archived from the original on October 16, 2013. Retrieved October 15, 2013. ^ a b "Cray C and C++ Reference Manual (8.7) S-2179". Retrieved Jun 24, 2020. ^ "Abstract Data Models - Windows applications". May 30, 2018. ^ "ILP32 for AArch64 Whitepaper". ARM Limited. June 9, 2015. Archived from the original on December 30, 2018. Retrieved October 9, 2018. ^ "Apple devices in 2018". woachk, security researcher. October 6, 2018. 64-bit Transition Guide, Mac Developer Library Karpov, Andrey. "A Collection of Examples of 64-bit Errors in Real Programs". Archived from the original on September 23, 2021. Kilgard, Mark J. "Is your X code ready for 64-bit?". Archived from the original on June 3, 2001. Retrieved September 26, 2012. Lessons on development of 64-bit C/C++ applications at the Wayback Machine (archived April 14, 2021) 64-Bit Programming Models: Why LP64? AMD64 (EM64T) architecture Retrieved from " 3Computer architecture bit width "16-bit" redirects here. For the color encoding, see 16-bit color. For the video gaming era, see 16-bit era. For other uses, see 16-bit (disambiguation). This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources: "16-bit computing" – news · newspapers · books · scholar · JSTOR (March 2023) (Learn how and when to remove this message) Computer architecture bit widths Bit 1481216182426303132364548606412825651 2bit slicing Application 8163264 Binary floating-point precision 16 (×½)2432 (×1)4064 (×2)80128 (×4)256 (×8) Decimal floating-point precision 3264128 vte In computer architecture, 16-bit integers, memory addresses, or other data units are those that are 16 bits (2 octets) wide. Also, 16-bit central processing unit (CPU) and arithmetic logic unit (ALU) architectures are those that are based on registers, address buses, or data buses of that size. 16-bit microcomputers are microcomputers that use 16-bit microprocessors. A 16-bit register can store 216 different values. The range of integer values that can be stored in 16 bits depends on the integer representation used. With the two most common representations, the range is 0 through 65,535 (216 − 1) for representation as an (unsigned) binary number, and −32,768 (−1 × 215) through 32,767 (215 − 1) for representation as two's complement. Since 216 is 65,536, a processor with 16-bit memory addresses can directly access 64 KB (65,536 bytes) of byte-addressable memory. If a system uses segmentation with 16-bit segment offsets, more can be accessed. As of 2025, 16-bit microcontrollers cost well under a dollar (similar to close in price legacy 8-bit[1][2]); the cheapest 16-bit microcontrollers cost less than other types[3] including any 8-bit (and are more powerful, and easier to program generally), making 8-bit legacy, i.e. not worth it for new applications; 32-bit microcontrollers are also well under half a dollar, cheaper than most 16-bit, with few exceptions, then close to cheapest 16-bit. Digital Equipment Corporation PDP-11, a popular 16-bit minicomputer during the 1970s The MIT Whirlwind (c. 1951)[4][5] was quite possibly the first-ever 16-bit computer. It was an unusual word size for the era; most systems used six-bit character code and used a word length of some multiple of 6-bits. This changed with the effort to introduce ASCII, which used a 7-bit code and naturally led to the use of an 8-bit multiple which could store a single ASCII character or two binary-coded decimal digits. The 16-bit word length thus became more common in the 1960s, especially on minicomputer systems. Early 16-bit computers (c. 1965–70) include the IBM 1130,[6] the HP 2100,[7] the Data General Nova,[8] and the DEC PDP-11.[9] Early 16-bit microprocessors, often modeled on one of the mini platforms, began to appear in the 1970s. Examples (c. 1973–76) include the five-chip National Semiconductor IMP-16 (1973),[10] the two-chip NEC μCOM-16 (1974),[11][10] the three-chip Western Digital MCP-1600 (1975), and the five-chip Toshiba T-3412 (1976).[10] Early single-chip 16-bit microprocessors (c. 1975–76) include the Panafacom MN1610 (1975),[12][13][10] National Semiconductor PACE (1975), General Instrument CP1600 (1975), Texas Instruments TMS9900 (1976),[10] Ferranti F100-L, and the HP BPC. Other notable 16-bit processors include the Intel 8086, the Intel 80286, the WDC 65C816, and the Zilog Z8000. The Intel 8088 was binary compatible with the Intel 8086, and was 16-bit in that its registers were 16 bits wide, and arithmetic instructions could operate on 16-bit quantities, even though its external bus was 8 bits wide. 16-bit processors have been almost entirely supplanted in the personal computer industry, and are used less than 32-bit (or 8-bit) CPUs in embedded applications. The Motorola 68000 is sometimes called 16-bit because of the way it handles basic arithmetic. The instruction set was based on 32-bit numbers and the internal registers were 32 bits wide, so by common definitions, the 68000 is a 32-bit design. Internally, 32-bit arithmetic is performed using two 16-bit operations, and this leads to some descriptions of the system as 16-bit, or "16/32". Such solutions have a long history in the computer field, with various designs performing math even one bit at a time, known as "serial arithmetic", while most designs by the 1970s processed at least a few bits at a time. A common example is the Data General Nova, which was a 16-bit design that performed 16-bit math as a series of four 4-bit operations. 4-bits was the word size of a widely available single-chip ALU and thus allowed for inexpensive implementation. Using the definition being applied to the 68000, the Nova would be a 4-bit computer, or 4/16. Not long after the introduction of the Nova, a second version was introduced, the SuperNova, which included four of the 4-bit ALUs running in parallel to perform math 16 bits at a time and therefore offer higher performance. This was invisible to the user and the programs, which always used 16-bit instructions and data. In a similar fashion, later 68000-family members, starting with the Motorola 68020, had 32-bit ALUs. One may also see references to systems being, or not being, 16-bit based on some other measure. One common one is when the address space is not the same size of bits as the internal registers. Most 8-bit CPUs of the 1970s fall into this category; the MOS 6502, Intel 8080, Zilog Z80 and most others had 16-bit address space which provided 64 KB of address space. This also meant address manipulation required two instruction cycles. For this reason, most processors had special 8-bit addressing modes, the zero page, improving speed. This sort of difference between internal register size and external address size remained in the 1980s, although often reversed, as memory costs of the era made a machine with 32-bit addressing, 2 or 4 GB, a practical impossibility. For example, the 68000 exposed only 24 bits of addressing on the DIP, limiting it to a still huge (for the era) 16 MB.[14] A similar analysis applies to Intel's 80286 CPU replacement, called the 386SX, which is a 32-bit processor with 32-bit ALU and internal 32-bit data paths with a 16-bit external bus and 24-bit addressing of the processor it replaced. In the context of IBM PC compatible and Wintel platforms, a 16-bit application is any software written for MS-DOS, OS/2 1.x or early versions of Microsoft Windows which originally ran on the 16-bit Intel 8088 and Intel 80286 microprocessors. Such applications used a 20-bit or 24-bit segment or selector-offset address representation to extend the range of addressable memory locations beyond what was possible using only 16-bit addresses. Programs containing more than 216 bytes (65,536 bytes) of instructions and data therefore required special instructions to switch between their 64-kilobyte segments, increasing the complexity of programming 16-bit applications. This list is incomplete; you can help by adding missing items. (November 2021) Angstrem 1801 series CPU Data General Nova Eclipse Digital Equipment Corporation PDP-11 (for LSI-11, see Western Digital, below) DEC J-11 DEC T-11 EnSilica eSi-1600 Fairchild Semiconductor 9440 MICROFLAME Ferranti Ferranti F100-L Ferranti F200-L General Instrument CP1600 Hewlett-Packard HP 21xx/2000/1000/98xx/BPC HP 3000 Honeywell Honeywell Level 6/DPS 6 IBM 1130/1800 System/7 Series/1 System/36 Infineon XE166 family C166/C167 family XC2000 Intel Intel 8086/Intel 8088 Intel 80186/Intel 80188 Intel 80286 Intel MCS-96 Lockheed MAC-16 MIL-STD-1750A Motorola Motorola 68HC12 Motorola 68HC16 National Semiconductor IMP-16 PACE/INS8900 NEC μCOM-16 NEC V20 and V30 Panafacom MN1610 Renesas Renesas M16C [jp] (16-bit registers, 24-bit address space) Ricoh Ricoh 5A22 (WDC 65816 clone used in SNES) Texas Instruments Texas Instruments TMS9900 TI MSP430 Toshiba T-3412 Western Design Center WDC 65816/65802 Western Digital MCP-1600 used in the DEC LSI-11 used in the Pascal MicroEngine used in the WD16 Xerox Alto Zilog Zilog Z8000 Zilog Z280 Microprocessor § 16-bit designs Influence of the IBM PC on the personal computer market § Before the IBM PC's introduction 74181 (key component of some early 16-bit and other CPUs) Audio bit depth – as 16-bit is the most common bit depth used, e.g. on CD audio. ^ "The Amazing $1 Microcontroller". Jay Carlson. Retrieved 2025-05-21. ^ "STM8S103K3T6CTR 28 1KB STM8 8KB LQFP-32(7x7) Microcontroller". ^ "16-bit Microcontrollers - MCU". ^ "Year 1951". Computer History Museum. (see also "Year 1943".). ^ Digital Press, Digital at Work Archived 2013-07-02 at the Wayback Machine, Pearson, 1992, ISBN 1-55558-092-0, pp. 4, 23. ^ "The IBM 1130 computing system". IBM Archives. Archived from the original on March 18, 2005. ^ "HP 2116". Computer History Museum. ^ "Data General Nova minicomputer". Computer History Museum. Archived from the original on 2013-05-17. Retrieved 2012-06-11. ^ Pearson, Jamie Parker (September 1992). Digital at work: snapshots from the first thirty-five years. Digital Press. pp. 58–61. ISBN 978-1-55558-092-6. ^ a b c d e Belzer, Jack; Holzman, Albert G.; Kent, Allen (1978). Encyclopedia of Computer Science and Technology. Vol. 10 – Linear and Matrix Algebra to Microorganisms: Computer-Assisted Identification. CRC Press. p. 402. ISBN 9780824722609. ^ "1970s: Development and evolution of microprocessors" (PDF). Semiconductor History Museum of Japan. Archived from the original (PDF) on 2019-06-27. Retrieved 2019-06-27. ^ "16-bit Microprocessors". CPU Museum. Retrieved 5 October 2010. ^ "History". PFU. Retrieved 5 October 2010. ^ Motorola M68000 Family, Programmer's Reference Manual (PDF). Motorola, Inc. 1992. sec. 2.4, pp. 2–21. Retrieved 2023-06-05. Retrieved from " 4 The following pages link to 16-bit computing External tools (link count transclusion count sorted list) · See help page for transcluding these entries Showing 50 items. View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)ASCII (links | edit) Atanasoff–Berry computer (links | edit) Apple II (links | edit) Atari ST (links | edit) AMD (links | edit) Athlon (links | edit) Accumulator (computing) (links | edit) Advanced Power Management (links | edit) Central processing unit (links | edit) Control unit (links | edit) Control store (links | edit) Processor design (links | edit) Complex instruction set computer (links | edit) Digital Equipment Corporation (links | edit) Data General Nova (links | edit) DEC Alpha (links | edit) Execution unit (links | edit) Generation X (links | edit) Game Boy Advance (links | edit) History of the graphical user interface (links | edit) Intel (links | edit) Intel 80286 (links | edit) Intel 8080 (links | edit) Intel 8086 (links | edit) Intel 8088 (links | edit) I386 (links | edit) Instruction register (links | edit) Integrated circuit (links | edit) I486 (links | edit) I486SX (links | edit) Intel 80186 (links | edit) Intel DX4 (links | edit) Itanium (links | edit) AMD K6 (links | edit) Microsoft (links | edit) Marilyn Manson (band) (links | edit) Microprocessor (links | edit) Microcode (links | edit) Microassembler (links | edit) MIPS architecture (links | edit) Motorola 68000 (links | edit) MOS Technology 6502 (links | edit) Motorola 6809 (links | edit) Motorola 68HC11 (links | edit) Instructions per second (links | edit) Microcontroller (links | edit) Nintendo (links | edit) Nintendo 64 (links | edit) Amiga Original Chip Set (links | edit) PDP-10 (links | edit) View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500) Retrieved from " WhatLinksHere/16-bit_computing"