



## Arm instruction set

Family of computer architectures "ARM architecture" redirects here. For the Australian architectural firm, see ARM Architecture (company). ARMDesigner Sophie Wilson Steve Furber Acorn Computers/Arm Holdings Bits32-bit, 64-bitIntroduced1985; 40 years ago (1985)DesignRISCTypeLoad-storeBranchingCondition code, compare and branchOpenProprietary ARM AArch64 (64/32-bit)Introduced2011; 14 years ago (2011)VersionARMv8.7-A, ARMv8.9-A, ARMv8.9-A, ARMv8.9-A, ARMv8.9-A, ARMv8.9-A, ARMv8.2-A, ARMv8.2-A, ARMv9.2-A, ARMv9.2-A, ARMv9.2-A, ARMv9.2-A, ARMv9.2-A, ARMv9.2-A, ARMv9.2-A, ARMv8.9-A, ARMv8.9-A AArch32/A32 use 32-bit instructions, AArch32/T32 (Thumb-2) uses mixed 16- and 32-bit instructions[1]EndiannessBi (little as default)ExtensionsSVE, SVE2, SME, AES, SM3, SM4, SHA, CRC32, RNDR, TME; All mandatory: Thumb-2, Neon, VFPv4-D16, VFPv4; obsolete: Thumb and JazelleRegistersGeneral-purpose31 × 64-bit integer registers[1]Floating point32 × 128-bit registers[1] for scalar 32- and 64-bit FP or SIMD FP or integer; or cryptography ARM AArch32 (32-bit)VersionARMv9-R, ARMv7-A, default)ExtensionsThumb, Thumb-2, Neon, Jazelle, AES, SM3, SM4, SHA, CRC32, RNDR, DSP, Saturated, FPv5, Helium; obsolete since ARMv8: Thumb and JazelleRegisters, including R14 (link register), but not R15 (PC)Floating pointUp to 32 × 64-bit registers, [2] SIMD/floating-point (optional) ARM 32-bit (legacy)VersionARMv6, ARMv5, ARMv47, ARMv3, ARMv2Encoding32-bit, except Thumb extensionsThumb, JazelleRegistersGeneral-purpose15 × 32-bit integer registers, including R14 (link register), but not R15 (PC, 26-bit addressing in older)Floating pointNone ARM (stylised in lowercase as arm, formerly an acronym for Advanced RISC Machines and originally Acorn RISC Machine) is a family of RISC instruction set architectures (ISAs) for computer processors. Arm Holdings develops the ISAs and licenses them to other companies, who build the physical devices that use the instruction set. It also designs and licenses cores that implement these ISAs. Due to their low costs, low power consumption, and low heat generation, ARM processors are useful for light, portable, battery-powered devices, including smartphones, laptops, and tablet computers, as well as embedded systems.[3][4][5] However, ARM processors are also used for desktops and servers, including Fugaku, the world's fastest supercomputer from 2020[6] to 2022. With over 230 billion ARM chips produced, [7][8] since at least 2003, and with its dominance increasing every year[update], ARM is the most widely used family of instruction set architectures. [9][4][10][11][12] There have been several generations of the ARM design. The original ARM1 used a 32-bit internal structure but had a 26-bit address space that limited it to 64 MB of main memory. This limitation was removed in the ARMv3 series, which has a 32-bit address space, and several additional generations up to ARMv7 remained 32-bit. Released in 2011, the ARMv8-A architecture added support for a 64-bit address space and 64-bit arithmetic with its new 32-bit fixed-length instruction set. [13] Arm Holdings has also released a series of additional instruction sets for different roles: the "Thumb" extensions add both 32- and 16-bit instructions for improved code density, while Jazelle added instructions for directly handling Java bytecode. More recent changes include the addition of simultaneous multithreading (SMT) for improved performance or fault tolerance.[14] Main article: BBC Micro, introduced in December 1981. This was a relatively conventional machine based on the MOS Technology 6502 CPU but ran at roughly double the performance of competing designs like the Apple II due to its use of faster dynamic random-access memory (DRAM). Typical DRAM of the era ran at about 2 MHz; Acorn arranged a deal with Hitachi for a supply of faster 4 MHz parts.[15] Machines of the era generally shared memory between the processor and the framebuffer, which allowed the processor to quickly update the contents of the screen without having to perform separate input/output (I/O). As the timing of the video hardware had to have priority access to that memory. Due to a quirk of the 6502's design, the CPU left the memory untouched for half of the time. Thus by running the CPU at 1 MHz, the video system could read data during those down times, taking up the total 2 MHz bandwidth of the RAM. In the BBC Micro, the use of 4 MHz RAM allowed the same technique to be used, but running at twice the speed. This allowed it to outperform any similar machine on the market.[16] Main article: Acorn Business Computer 1981 was also the year that the IBM Personal Computer was introduced. Using the recently introduced Intel 8088, a 16-bit CPU compared to the 6502's 8-bit design, it offered higher overall performance. Its introduced Intel 8088, a 16-bit CPU compared to the 6502's 8-bit design. prior five years began to change to a must-have business tool where the earlier 8-bit designs simply could not compete. Even newer 32-bit designs were also coming to market, such as the Motorola 68000[17] and National Semiconductor NS32016.[18] Acorn began considering how to compete in this market and produced a new paper design named the Acorn Business Computer. They set themselves the goal of producing a machine with ten times the performance of the BBC Micro, but at the same time, the recent introduction of the Apple Lisa brought the graphical user interface (GUI) concept to a wider audience and suggested the future belonged to machines with a GUI.[20] The Lisa, however, cost \$9,995, as it was packed with support chips, large amounts of memory, and a hard disk drive, all very expensive then.[21] The engineers then began studying all of the CPU designs available. Their conclusion about the existing 16-bit designs was that they were a lot more expensive and were still "a bit crap",[22] offering only slightly higher performance than their BBC Micro design. They also considered the new 32-bit designs, but these cost even more and had the same issues with support chips.[23] According to Sophie Wilson, all the processors tested at that time performed about the same, with about a 4 Mbit/s bandwidth.[24][a] Two key events led Acorn down the path to ARM. One was the publication of a series of reports from the University of California, Berkeley, which suggested that a simple chip design could nevertheless have extremely high performance, much higher than the latest 32-bit designs on the market. [25] The second was a visit by Steve Furber and Sophie Wilson to the Western Design Center, a company run by Bill Mensch and his sister, which had become the logical successor to the MOS team and was offering new versions like the WDC 65C02. The Acorn team saw high school students producing chip layouts on Apple II machines, which suggested that anyone could do it.[26][27] In contrast, a visit to another design firm working on modern 32-bit CPU revealed a team with over a dozen members who were already on revision H of their design and yet it still contained bugs.[b] This cemented their late 1983 decision to begin their own CPU designs, the Acorn RISC Machine.[28] The original Berkeley RISC designs were in some sense teaching systems, not designed specifically for outright performance. To the RISC's basic register-heavy and load/store concepts, ARM added a number of the well-received design notes of the 6502. Primary among them was the ability to quickly serve interrupts, which allowed the machines to offer reasonable input/output performance with no added external hardware. To offer interrupts with similar performance as the 6502, the ARM design limited its physical address space to 64 MB of total addressable space, requiring 26 bits of an instruction address. As instructions were a bytes (32 bits) long, and required to be aligned on 4-byte boundaries, the lower 2 bits of an instruction address were always zero. This meant the program counter (PC) only needed to be 24 bits, allowing it to be stored along with the eight bit processor flags in a single 32-bit register. That meant that upon receiving an interrupt, the entire machine state could be saved in a single operation, whereas had the PC been a full 32-bit value, it would require separate operations to store the PC and the status flags. This decision halved the interrupt overhead. [29] Another change, and among the most important in terms of practical real-world performance, was the modification of the instruction set to take advantage of page mode allowed subsequent accesses of memory to run twice as fast if they were roughly in the same location, or "page", in the DRAM chip. Berkeley's design did not consider page mode and treated all memory equally. The ARM design added special vector-like memory access instructions, the "S-cycles", that could be used to fill or save multiple registers in a single page mode. This doubled memory performance when they could be used to fill or save multiple registers in a single page mode. designs used register windows to reduce the number of register saves and restores performed in procedure calls; the ARM design did not adopt this. Wilson developed the instruction set, writing a simulation of the processor in BBC BASIC that ran on a BBC Micro with a second 6502 processor.[31][32] This convinced Acorn engineers they were on the right track. Wilson approached Acorn's CEO, Hermann Hauser, and requested more resources. Hauser gave his approval and assembled a small team to design the actual processor based on Wilson's ISA.[33] The official Acorn RISC Machine project started in October 1983. ARM1 2nd processor for the BBC Micro Acorn chose VLSI Technology as the "silicon partner", as they were a source of ROMs and custom chips for Acorn. Acorn provided the layout and production. The first samples of ARM silicon worked properly when first received and tested on 26 April 1985.[3] Known as ARM1, these versions ran at 6 MHz.[34] The first ARM application was a second processor for the BBC Micro, where it helped in developing simulation software to finish development of the support chips (VIDC, IOC, MEMC), and sped up the CAD software used in ARM2 development. Wilson subsequently rewrote BBC BASIC in ARM assembly language. The in-depth knowledge gained from designing the instruction set enabled the code to be very dense, making ARM BBC BASIC an extremely good test for any ARM emulator. The result of the simulations on the ARM1 boards led to the late 1986 introduction of the ARM2 design running at 8 MHz, and the early 1987 speed-bumped version at 10 to 12 MHz.[c] A significant change in the underlying architecture was the addition of a Booth multiplier, whereas formerly multiplication had to be carried out in software.[36] Further, a new Fast Interrupt reQuest mode, FIQ for short, allowed registers 8 through 14 to be replaced as part of the interrupt itself. This meant FIQ requests did not have to save out their registers, further speeding interrupts.[37] The first use of the ARM2 were in ARM Evaluations systems, supplied as a second processor for BBC Micro and Master machines, [39] and the Acorn Archimedes personal computer models A305, A310, and A440, launched on the 6th June 1987. According to the Dhrystone benchmark, the ARM2 was roughly seven times the performance of a typical 7 MHz 68000-based system like the Amiga or Macintosh SE. It was twice as fast as an Intel 80386 running at 16 MHz, and about the same speed as a multi-processor VAX-11/784 superminicomputer. The only systems that beat it were the Sun SPARC and MIPS R2000 RISC-based workstations.[40] Further, as the CPU was designed for high-speed I/O, it dispensed with many of the support chips seen in these machines; notably, it lacked any dedicated direct memory access (DMA) controller which was often found on workstations. The graphics system was also simplified based on the same set of underlying assumptions about memory and timing. The result was a dramatically simplified design, offering performance on par with expensive workstations but at a price point similar to contemporary desktops. [40] The ARM2 featured a 32-bit data bus, 26-bit address space and 27 32-bit registers, of which 16 are accessible at any one time (including the PC). [41] The ARM2 had a transistor count of just 30,000 [42] compared to Motorola's six-year-older 68000 model with around 68,000. Much of this simplicity came from the lack of microcode, which represents about one-quarter to one-third of the 68000's transistors, and the lack of microcode, which represents about one-quarter to a low power consumption and simpler thermal packaging by having fewer powered transistors. Nevertheless, ARM2 offered better performance than the contemporary 1987 IBM PS/2 Model 50, which initially utilised an Intel 80286, offering 1.8 MIPS @ 10 MHz, and later in 1987, the 2 MIPS of the PS/2 70, with its Intel 386 DX @ 16 MHz.[43][44] A successor, ARM3, was produced with a 4 KB cache, which further improved performance.[45] The address bus was extended to 32 bits in the ARM6, but program code still had to lie within the first 64 MB of memory in 26-bit compatibility mode, due to the reserved bits for the status flags.[46] Microprocessor-based system on a chip Die of an ARM610 microprocessor In the late 1980s, Apple Computer and VLSI Technology started working with Acorn on newer versions of the ARM core. In 1990, Acorn spun off the design team into a new company, Arm Holdings plc, floated on the London Stock Exchange and Nasdaq in 1998.[50] The new Apple-ARM work would eventually evolve into the ARM6, first released in early 1992. Apple used the ARM610 as the basis for their Apple Newton PDA. In 1994, Acorn used the ARM610 as the main central processing unit (CPU) in their RiscPC computers. DEC licensed the ARMv4 architecture and produced the StrongARM.[51] At 233 MHz, this CPU drew only one watt (newer versions draw far less). This work was later passed to Intel as part of a lawsuit settlement, and Intel took the opportunity to supplement their i960 line with the StrongARM. Intel later developed its own high performance implementation named XScale, which it has since sold to Marvell. Transistor count of the ARM core remained essentially the same throughout these changes; ARM2 had 30,000 transistors, [52] while ARM6 grew only to 35,000. [53] In 2010, producers of chips based on ARM architectures reported shipments of 6.1 billion ARM-based processors. representing 95% of smartphones, 35% of digital televisions and set-top boxes, and 10% of mobile computers. In 2011, the 32-bit ARM architecture was the most popular 32-bit ARM architecture in mobile devices and the most popular 32-bit ARM architecture was the most popular 32-bit ARM architecture was the most popular 32-bit ARM architecture in mobile devices and the most popular 32-bit ARM architecture was the most popular percent of the world's mobile devices".[57] See also: Arm Holdings's primary business is selling IP cores, which licensees use to create microcontrollers (MCUs), CPUs, and systems-on-chips based on those cores. The original design manufacturer combines the ARM core with other parts to produce a complete device, typically one that can be built in existing semiconductor fabrication has been the ARM7TDMI with hundreds of millions sold. Atmel has been a precursor design center in the ARM7TDMI-based embedded system. The ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, PDAs and other mobile devices range from ARMv5 to ARMv8-A. In 2009, some manufacturers introduced netbooks based on ARM architectures used in smartphones, variety of licensing terms, varying in cost and deliverables. Arm Holdings provides to all licensees an integratable hardware development toolset (compiler, debugger, software development kit), and the right to sell manufactured silicon containing the ARM CPU. SoC packages integrating ARM's core designs include Nvidia Tegra's first three generations, CSR plc's Quatro family, ST-Ericsson's Nova and NovaThor, Silicon Labs's Precision32 MCU, Texas Instruments's OMAP products, Apple's A4, A5, and A5X, and NXP's i.MX. Fabless licensees, who wish to integrate an ARM core into their own chip design, are usually only interested in acquiring a ready-to-manufacture verified semiconductor intellectual property core. For these customers, Arm Holdings delivers a gate netlist description of the chosen ARM core, along with an abstracted simulation model and test programs to aid design integration. More ambitious customers, including integrated device manufacturers (IDM) and foundry operators, choose to acquire the processor IP in synthesizable RTL (Verilog) form. With the synthesizable RTL, the customer has the ability to perform architectural level optimisations and extensions. This allows the designer to achieve exotic design goals not otherwise possible with an unmodified netlist (high clock speed, very low power consumption, instruction set extensions, etc.). While Arm Holdings does not grant the licensees may freely sell manufactured products such as chip devices, evaluation boards and complete systems. Merchant foundries can be a special case; not only are they allowed to sell finished silicon containing ARM cores, they generally hold the right to re-manufacture ARM cores for other customers. Arm Holdings prices its IP based on perceived value. Lower performing Cores for other customers, a synthesisable core costs more than a hard macro (blackbox) core. Complicating price matters, a merchant foundry that holds an ARM licence, such as Samsung or Fujitsu, can offer fab customers reduced or eliminate payment of ARM's upfront licence fee. Compared to dedicated semiconductor foundries (such as TSMC and UMC) without in-house design services, Fujitsu/Samsung charge two- to three-times more per manufactured wafer.[citation needed] For low to mid volume applications, a design service foundry offers lower overall pricing (through subsidisation of the licence fee). For high volume mass-produced parts, the long term cost reduction achievable through lower wafer pricing reduces the impact of ARM's NRE (non-recurring engineering) costs, making the dedicated foundry a better choice. Companies that have developed chips with cores designed by Arm include Amazon.com's Annapurna Labs subsidiary,[59] Analog Devices, Apple, AppliedMicro (now: MACOM Technology Solutions[60]), Atmel, Broadcom, Cavium, Cypress Semiconductor, Freescale Semiconductor February 2016, ARM announced the Built on ARM Cortex Technology licence, often shortened to Built on Cortex (BoC) licence. This licence allows companies to partner with ARM and make modifications to ARM Cortex designs. These design modifications will not be shared with other companies. freedom, for example Kryo 280. Companies that are current licensees of Built on ARM Cortex Technology include Qualcomm.[61] Companies that have designed cores that implement an ARM architecture include Apple, AppliedMicro (now: Ampere Computing), Broadcom, Cavium (now: Marvell), Digital Equipment Corporation, Intel, Nvidia, Qualcomm in 2021). On 16 July 2019, ARM announced ARM Flexible Access. ARM Flexible Access provides unlimited access to included ARM intellectual property (IP) for development. Per product licence fees are required once a customer reaches foundry tapeout or prototyping. [62][63] 75% of ARM's most recent IP over the last two years are included in ARM Flexible Access. As of October 2019: CPUs: Cortex-A32, Co A34, Cortex-A35, Cortex-A53, Cortex-A53, Cortex-R5, Cortex-R5, Cortex-R5, Cortex-M0, Cortex-M0, Cortex-M0, Cortex-M3, Cortex-M4, Cor AMBA, XHB-400 AXI-AHB System Controllers: CoreLink GIC-500, PL192 VIC, BP141 TrustZone Memory Wrapper, CoreLink TZC-400, CoreLink MMU-500, BP140 Memory Interface Security IP: CryptoCell-312, PL022 SPI, PL031 RTC Debug & Trace CoreSight SDC-600, CoreSight SDC-600, CoreSight SDC-600, CoreSight Structure Controller Design Kits: Corstone-201 Physical IP: Artisan PIK for Cortex-M33 TSMC 22ULL including memory compilers, logic libraries, GPIOs and documentation Tools & Materials: Socrates IP ToolingARM Design Studio, Virtual System Models Support: Standard ARM Technical support; ARM online training and design reviews Main article: List of ARM processors Architecture Corebit-width Cores Profile References Arm Ltd. Third-party ARMv1 32 ARM1 Classic [a 1] ARMv2 32 ARM2, ARM250, ARM3 Amber, STORM Open Soft Core[64] Classic [a 2] ARMv4 32 ARM6, ARM7 Classic [a 2] ARMv4 32 ARM8 StrongARM, FA526, ZAP Open Source Processor Core Classic [a 2] ARMv5TE 32 ARM7EJ, ARM9E, ARM10E XScale, FA626TE, Feroceon, PJ1/Mohawk Classic ARMv6 32 ARM Cortex-M0, ARM Cortex-M0, ARM Cortex-M0, ARM Cortex-M0, ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor Microcontroller ARMv7-M 32 ARM Cortex-M1, SecurCore SC300 Apple M7 motion coprocessor M1 M23,[66] ARM Cortex-M33[67] Microcontroller [68] ARMv8.1-M 32 ARM Cortex-R5, ARM Cortex-A8, ARM Cortex-A9, ARM Cortex-A12, ARM Cortex-A15, ARM Cortex-A15, ARM Cortex-A35, [74] ARM Cortex-A35, [74] ARM Cortex-A32, [75] ARM Cortex-A32, [76] ARM Cortex-A32, [77] X-Gene, Nvidia Denver 1/2, Cavium Cortex-A75,[86] ARM Cortex-A76,[87] ARM Cortex-A77, ARM Cortex-A77, ARM Cortex-A65AE[91] (also having e.g. ARMv8.4 Dot Product; made for safety critical tasks such as advanced driver-assistance systems (A12, A12X, A12Z), Marvell ThunderX3 (v8.3+)[92] Application 64 ARM Neoverse V1 Apple Lightning+Thunder (A13), Apple Firestorm+Icestorm (A14, M1) Application ARMv8.5-A 64/32 Application ARMv8.6-A 64 Application ARMv8.6-A 64 Apple Coll (A17), Apple Everest+Sawtooth (A16),[93] Apple Coll (A17),[93] Apple Coll (A17),[93] Apple Coll (A17),[93] Apple Coll (A17),[93] Apple Coll (A17),[ ARM Cortex-A510, ARM Cortex-A710, ARM Cortex-A715, ARM Cortex-X2, ARM Cortex-X2, ARM Neoverse E2, ARM Neoverse Donan/BravaChop/Brava (Apple M4),[100] Apple Tupai/Tahiti (A18) Application [101] ARMv9.3-A 64 TBA Application [101] ARMv9.4-A 64 TBA Application [101] ARMv9.4-A 64 TBA Application [102] ARMv9.5-A 64 TBA Application [102] ARMv9.5-A 64 TBA Application [103] ARMv9.6-A 64 TBA Application [103] ARMv9.4-A 64 TBA Application [103] ARMv9.6-A 64 TBA Application [103] ARMv limited to 26 bits; with upper bits, then, used for status flags in the program counter register. ^ a b c ARMv3 included a compatibility mode optional in ARMv4, and removed entirely in ARMv5. Arm provides a list of vendors who implement ARM cores in their design (application specific standard products (ASSP), microprocessor and microcontrollers).[105] Tronsmart MK908, a Rockchip-based quad-core Android "mini PC", with a microSD card next to it for a size comparison Main article: List of products using ARM processors ARM cores are used in a number of products, particularly PDAs and smartphones. Some computing examples are Microsoft's first generation Surface, Surface 2 and Pocket PC devices (following 2002), Apple's iPhone smartphones and iPod portable media players, Canon PowerShot digital cameras, Nintendo Switch hybrid, the Wii security processor and 3DS handheld game consoles, and TomTom turn-by-turn navigation systems. In 2005, Arm took part in the development of Manchester University's computer SpiNNaker, which used ARM cores to simulate the human brain.[106] ARM chips are also used in Raspberry Pi, BeagleBoard, BeagleBone, PandaBoard, and other single-board computers, because they are very small, inexpensive, and consume very little power. An ARMv7 was used to power the CuBox family of single-board computers. See also: Comparison of ARMv7-A processors The 32-bit ARM architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), was the most widely used architecture (ARM32), such as ARMv7-A (implementing AArch32; see section on Armv8-A for more on it), see section on Armv the primary source of documentation on the ARM processor architecture and instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are required to support (such as instruction set, distinguishing interfaces that all ARM processors are r architecture "profiles": A-profile, the "Application" profile, implemented by 32-bit cores in the Cortex-R series Although the architecture profiles were first defined for ARMv7, ARM subsequently defined the ARMv6-M architecture (used by the Cortex M0/M0+/M1) as a subset of the ARMv7-M profile, the 32-bit ARM architecture specifies several CPU modes, depending on the implemented architecture features. At any moment in time, the CPU can be in only one mode, but it can switch modes due to external events (interrupts) or programmatically.[107] User mode: A privileged mode that is entered whenever the processor accepts a fast interrupt. Supervisor (svc) mode: A privileged mode entered whenever a prefetch abort or data abort exception occurs. Undefined mode: A privileged mode that is entered whenever an undefined instruction exception occurs. System mode (ARMv4 and above): The only privileged mode that is not entered by an exception. It can only be entered by executing an instruction that explicitly writes to the mode is of the Current Program Status Register (CPSR) from another privileged mode is introduced to support TrustZone extension in ARM cores. Hyp mode (ARMv7 Virtualization Extensions, ARMv8 EL2): A hypervisor mode that supports Popek and Goldberg virtualization requirements for the non-secure operation of the CPU.[108][109] Thread mode (ARMv7-M, ARMv8-M): A mode which can be specified as either privileged or unprivileged. Whether the Main Stack Pointer (MSP) or Process Stack Pointer (PSP) is used can also be specified in CONTROL register with privileged access. This mode is designed for user tasks in RTOS environment but it is typically used in bare-metal for super-loop. Handler mode (ARMv6-M, ARMv7-M, ARMv8-M): A mode dedicated for exception handling (except the RESET which are handled in Thread mode). Handler mode always uses MSP and works in privileged level. The original (and subsequent) ARM implementation was hardwired without microcode, like the much simpler 8-bit 6502 processor used in prior Acorn microcomputers. The 32-bit ARM architecture (and the 64-bit architecture for the most part) includes the following RISC features: Load-store architecture. No support for unaligned memory accesses in the original versions, support unaligned accesses for half-word load/store instructions with some limitations, such as no guaranteed atomicity.[110][111] Uniform 16 × 32-bit register file (including the program counter, stack pointer and the link register). Fixed instruction set added 16-bit instructions and increased code density. Mostly single clock-cycle execution. To compensate for the simpler design, compared with processors like the Intel 80286 and Motorola 68020, some additional design features were used: Conditional desi when desired. 32-bit barrel shifter can be used without performance penalty with most arithmetic instructions and address calculations. Has powerful indexed addressing modes. A link register supports fast leaf function calls. A simple, but fast, 2-priority-level interrupt subsystem has switched register banks. ARM includes integer arithmetic operations for add, subtract, and multiply; some versions of the architecture also support divide operations. ARM supports 32-bit × 32-bit multiplies. The divide instructions are only included in the following ARM architectures: Armv7-M and Armv7E-M architectures always includes the divide instructions. The instructions might not be implemented only in the Thumb instruction set, or implemented in both the Thumb and ARM instruction sets, or implemented if the Virtualization Extensions are included.[114] Registers across CPU modes usr sys svc abt und irq fiq R0 R1 R2 R3 R4 R5 R6 R7 R8 R8 fiq R9 R9 fiq R10 R10 fiq R11 R11 fiq R12 R12 fig R13 R13 svc R13 abt R13 und R13 irq R14 fig R14 R14 svc R14 abt R14 und R14 irq R14 fig R15 CPSR spSR svc SPSR abt SPSR irq SPSR fig Registers R0 through R12 are the same across all CPU modes except FIQ mode. FIQ mode has its own distinct R8 through R12 registers. R13 and R14 are banked across all privileged CPU modes except system mode. That is, each mode that can be entered because of an exception has its own R13 and R14. These registers generally contain the stack pointer and the return address from function calls, respectively. Aliases: R13 is also referred to as SP, the stack pointer. R14 is also referred to as LR, the link register. R15 is also referred to as PC, the program counter. The Current Program counter. The Status Register (CPSR) has the FIQ disable bit. I (bit 7) is the IRQ disable bit. A (bit 8) is the imprecise data abort disable bit. E (bit 9) is the data endianness bit. IT (bits 10-15 and 25-26) is the if-then state bits. J (bit 24) is the Java state bit. Z (bit 27) is the sticky overflow bit. V (bit 28) is the carry/borrow/extend bit. Z (bit 30) is the zero bit. N (bit 31) is the negative/less than bit. Almost every ARM instruction has a conditional execution, one of the four-bit codes causes the instruction to be always executed. Most other CPU architectures only have condition codes on branch instructions.[116] Though the predicate takes up four of the 32 bits in an instruction code, and thus cuts down significantly on the encoding bits available for displacements in memory access instructions, it avoids branch instructions when generating code for small if statements. Apart from eliminating the branch instructions themselves, this preserves the fetch/decode/execute pipeline at the cost of only one cycle per skipped instruction. An algorithm for computing the greatest common divisor. In the C programming language, the algorithm can be written as: int gcd(int a, int b) { while  $(a \mid = b) / We enter the loop when a < b or a > b, but not when a == b if <math>(a > b) / When a < b we do that (no "if <math>(a < b)$ " needed since a != b is checked in while condition) b -= a: return a: } The same algorithm can be rewritten in a way closer to target ARM instructions as: loop: // Compare a and b GT = a > b; LT = a < b; NE = a != b; // Perform operations based on flag results if (GT) a -= b; // Subtract \*only\* if less-than if (NE) goto loop; // Loop \*only\* if compared values were not equal return a; and coded in assembly language as: ; assign a to register r0, b to r1 loop: CMP r0, r1; set condition "NE" if ( $a \neq b$ ), ; "GT" if (a > b), ; or "LT" if (a < b) SUBGT r0, r0, r1; if "GT" (Less Than), then a = a - b SUBLT r1, r1, r0; if "LT" (Less Than), then b = b - a BNE loop ; if "NE" (Not Equal), then loop B lr; return which avoids the branches around the then and else clauses. If r0 and r1 are equal then neither of the SUB instructions will be executed, eliminating the need for a conditional branch to implement the while check at the top of the loop, for example had SUBLE (less than or equal) been used. One of the ways that Thumb code provides a more dense encoding is to remove the four-bit selector from non-branch instructions. Another feature of the instruction set is the ability to fold shifts and rotates into the data processing (arithmetic, logical, and register-register move) instructions, so that, for example, the statement in C language: a += (j