**Лекция 2**

**Архитектура ARM** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *Advanced RISC Machine* — усовершенствованная [RISC](https://ru.wikipedia.org/wiki/RISC" \o "RISC)-машина; иногда — *[Acorn](https://ru.wikipedia.org/wiki/Acorn_Computers" \o "Acorn Computers) RISC Machine*) — [система команд](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4" \o "Система команд) и семейство описаний и готовых топологий 32-битных и 64-битных [микропроцессорных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80" \o "Микропроцессор)/[микроконтроллерных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80" \o "Микроконтроллер) [ядер](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0" \o "Ядро микропроцессора), разрабатываемых компанией [ARM Limited](https://ru.wikipedia.org/wiki/ARM_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "ARM (компания))[[1]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-1).

Среди [лицензиатов](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%B3%D0%BE%D0%B2%D0%BE%D1%80" \o "Лицензионный договор) готовых топологий ядер ARM — компании [AMD](https://ru.wikipedia.org/wiki/Advanced_Micro_Devices" \o "Advanced Micro Devices), [Apple](https://ru.wikipedia.org/wiki/Apple" \o "Apple), [Analog Devices](https://ru.wikipedia.org/wiki/Analog_Devices" \o "Analog Devices), [Atmel](https://ru.wikipedia.org/wiki/Atmel" \o "Atmel), [Xilinx](https://ru.wikipedia.org/wiki/Xilinx" \o "Xilinx), [Cirrus Logic](https://ru.wikipedia.org/wiki/Cirrus_Logic" \o "Cirrus Logic), [Intel](https://ru.wikipedia.org/wiki/Intel" \o "Intel) (до 27 июня 2006 года), [Marvell](https://ru.wikipedia.org/wiki/Marvell_Technology_Group" \o "Marvell Technology Group), [NXP](https://ru.wikipedia.org/wiki/NXP_Semiconductors" \o "NXP Semiconductors), [STMicroelectronics](https://ru.wikipedia.org/wiki/STMicroelectronics" \o "STMicroelectronics), [Samsung](https://ru.wikipedia.org/wiki/Samsung" \o "Samsung), [LG](https://ru.wikipedia.org/wiki/LG_Group" \o "LG Group), [MediaTek](https://ru.wikipedia.org/wiki/MediaTek" \o "MediaTek), [Qualcomm](https://ru.wikipedia.org/wiki/Qualcomm" \o "Qualcomm), [Sony](https://ru.wikipedia.org/wiki/Sony" \o "Sony), [Texas Instruments](https://ru.wikipedia.org/wiki/Texas_Instruments" \o "Texas Instruments), [Nvidia](https://ru.wikipedia.org/wiki/Nvidia" \o "Nvidia), [Freescale](https://ru.wikipedia.org/wiki/Freescale_Semiconductor" \o "Freescale Semiconductor), [Миландр](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BB%D0%B0%D0%BD%D0%B4%D1%80" \o "Миландр), ЭЛВИС[[2]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-2), [HiSilicon](https://ru.wikipedia.org/wiki/HiSilicon_Technologies" \o "HiSilicon Technologies), [Байкал электроникс](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D0%BA%D0%B0%D0%BB_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D1%81" \o "Байкал электроникс).

Значимые семейства процессоров: [ARM7](https://ru.wikipedia.org/wiki/ARM7" \o "ARM7), [ARM9](https://ru.wikipedia.org/wiki/ARM9" \o "ARM9), [ARM11](https://ru.wikipedia.org/wiki/ARM11" \o "ARM11) и [Cortex](https://ru.wikipedia.org/wiki/ARM_Cortex" \o "ARM Cortex)[[3]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-elec-2010-cortex-3)[[4]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-4).

Многие лицензиаты проектируют собственные топологии ядер на базе системы команд ARM: [DEC](https://ru.wikipedia.org/wiki/DEC" \o "DEC) [StrongARM](https://ru.wikipedia.org/wiki/StrongARM" \o "StrongARM), [Freescale](https://ru.wikipedia.org/wiki/Freescale" \o "Freescale) i.MX, [Intel XScale](https://ru.wikipedia.org/wiki/Intel_XScale" \o "Intel XScale), [NVIDIA Tegra](https://ru.wikipedia.org/wiki/NVIDIA_Tegra" \o "NVIDIA Tegra), [ST-Ericsson](https://ru.wikipedia.org/wiki/ST-Ericsson" \o "ST-Ericsson) [Nomadik](https://ru.wikipedia.org/w/index.php?title=Nomadik&action=edit&redlink=1" \o "Nomadik (страница отсутствует))[[англ.]](https://en.wikipedia.org/wiki/Nomadik), [Krait](https://ru.wikipedia.org/wiki/Krait_(CPU)" \o "Krait (CPU)) и [Kryo](https://ru.wikipedia.org/wiki/Kryo" \o "Kryo) в [Qualcomm](https://ru.wikipedia.org/wiki/Qualcomm" \o "Qualcomm) [Snapdragon](https://ru.wikipedia.org/wiki/Snapdragon" \o "Snapdragon), [Texas Instruments OMAP](https://ru.wikipedia.org/wiki/Texas_Instruments_OMAP" \o "Texas Instruments OMAP), [Samsung](https://ru.wikipedia.org/wiki/Samsung" \o "Samsung) [Hummingbird](https://ru.wikipedia.org/wiki/Exynos" \o "Exynos), [LG H13](https://ru.wikipedia.org/w/index.php?title=LG_H13&action=edit&redlink=1" \o "LG H13 (страница отсутствует)), [Apple](https://ru.wikipedia.org/wiki/Apple" \o "Apple) [A6](https://ru.wikipedia.org/wiki/Apple_Ax" \o "Apple Ax) и [HiSilicon K3](https://ru.wikipedia.org/wiki/HiSilicon_K3" \o "HiSilicon K3).

В 2006 году около 98 % из более чем миллиарда мобильных телефонов, продававшихся ежегодно, были оснащены, по крайней мере, одним процессором ARM[[5]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-Krazit-5). По состоянию на 2009 на процессоры ARM приходилось до 90 % всех встроенных 32-разрядных процессоров[[6]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-6). Процессоры ARM широко используются в потребительской электронике — в том числе смартфонах, мобильных телефонах и плеерах, портативных игровых консолях, калькуляторах, умных часах и компьютерных периферийных устройствах, таких, как жесткие диски или маршрутизаторы.

Эти процессоры имеют низкое энергопотребление, поэтому находят широкое применение во [встраиваемых системах](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Встраиваемая система) и преобладают на рынке мобильных устройств, для которых данный фактор критически важен.

**История**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=2" \o "Редактировать раздел \«История\») | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=2" \o "Редактировать код раздела \«История\»)]

После достижения некоторых успехов с компьютером [BBC Micro](https://ru.wikipedia.org/wiki/BBC_Micro" \o "BBC Micro) британская компания [Acorn Computers](https://ru.wikipedia.org/wiki/Acorn_Computers" \o "Acorn Computers) задумалась над переходом от относительно слабых процессоров [MOS Technology 6502](https://ru.wikipedia.org/wiki/MOS_Technology_6502" \o "MOS Technology 6502) к более производительным решениям и выходу на рынок бизнес-компьютеров с той же платформой [BBC Micro](https://ru.wikipedia.org/wiki/BBC_Micro" \o "BBC Micro). Такие процессоры, как [Motorola 68000](https://ru.wikipedia.org/wiki/Motorola_68000" \o "Motorola 68000) и 32016 от [National Semiconductor](https://ru.wikipedia.org/wiki/National_Semiconductor" \o "National Semiconductor), были для этого непригодны, а 6502 был недостаточно мощным, чтобы поддерживать [графический пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Графический пользовательский интерфейс)[[7]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-7).

Компании была нужна совершенно новая архитектура после того, как она протестировала все доступные ей процессоры и сочла их неэффективными. Acorn серьёзно настроился на разработку собственного процессора, и их инженеры начали изучать документацию [проекта RISC](https://ru.wikipedia.org/wiki/RISC" \l "RISC_%D0%B2_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8" \o "RISC), разработанного в [Университете Калифорнии в Беркли](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%9A%D0%B0%D0%BB%D0%B8%D1%84%D0%BE%D1%80%D0%BD%D0%B8%D0%B8_%D0%B2_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8" \o "Университет Калифорнии в Беркли). Они подумали, что раз уж группе студентов удалось создать вполне конкурентоспособный процессор, то их инженерам это будет несложно. Поездка в Western Design Center (Аризона) показала инженерам [Стиву Ферберу](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D1%80%D0%B1%D0%B5%D1%80,_%D0%A1%D1%82%D0%B8%D0%B2%D0%B5%D0%BD" \o "Фербер, Стивен) и [Софи Уилсон](https://ru.wikipedia.org/wiki/%D0%A3%D0%B8%D0%BB%D1%81%D0%BE%D0%BD,_%D0%A1%D0%BE%D1%84%D0%B8" \o "Уилсон, Софи) (на тот момент известной под именем Роджер[[8]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-8)), что им не потребуются невероятные ресурсы для осуществления этого плана.

Уилсон приступила к разработке [системы команд](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4" \o "Система команд), создавая симулятор нового процессора на компьютере BBC Micro. Её успехи в этом убедили инженеров Acorn, что они на верном пути. Но все же перед тем, как идти дальше, им требовалось больше ресурсов, настало время для Уилсон идти к директору Acorn [Герману Хаузеру](https://ru.wikipedia.org/w/index.php?title=%D0%A5%D0%B0%D1%83%D0%B7%D0%B5%D1%80,_%D0%93%D0%B5%D1%80%D0%BC%D0%B0%D0%BD&action=edit&redlink=1" \o "Хаузер, Герман (страница отсутствует)) и объяснить, в чём же дело. После того, как он дал добро, собралась небольшая команда для реализации модели Уилсон на аппаратном уровне.

**Acorn RISC Machine: ARM2**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=3" \o "Редактировать раздел \«Acorn RISC Machine: ARM2\») | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=3" \o "Редактировать код раздела \«Acorn RISC Machine: ARM2\»)]

Официальный проект Acorn RISC Machine был начат в октябре 1983 года. [VLSI Technology](https://ru.wikipedia.org/w/index.php?title=VLSI_Technology&action=edit&redlink=1" \o "VLSI Technology (страница отсутствует))[[англ.]](https://en.wikipedia.org/wiki/VLSI_Technology) была выбрана в качестве поставщика кремниевых компонентов, так как она уже снабжала Acorn микросхемами [ПЗУ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%BF%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE" \o "Постоянное запоминающее устройство) и некоторыми нестандартными интегральными схемами. Разработку возглавили Уилсон и [Фербер](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D1%80%D0%B1%D0%B5%D1%80,_%D0%A1%D1%82%D0%B8%D0%B2%D0%B5%D0%BD" \o "Фербер, Стивен). Их основной целью было достижение низкой [латентности](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D1%85%D0%B0%D1%80%D0%B0%D0%BA%D1%82%D0%B5%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8)" \o "Латентность (характеристика оперативной памяти)) обработки [прерывания](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Прерывание), как у MOS Technology 6502. Архитектура доступа к памяти, взятая от 6502, позволила разработчикам достичь хорошей [производительности](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0" \o "Производительность компьютера) без использования дорогостоящего в реализации модуля [DMA](https://ru.wikipedia.org/wiki/DMA" \o "DMA). Первый процессор был произведен VLSI 26 апреля 1985 года — именно тогда он впервые заработал и был назван ARM1. Первые серийные процессоры под названием ARM2 стали доступны в следующем году.

Его первое применение было в качестве второго процессора в [BBC Micro](https://ru.wikipedia.org/wiki/BBC_Micro" \o "BBC Micro), где он был использован при разработке программного обеспечения для моделирования, что позволило завершить работу над вспомогательными микросхемами компьютера, а также ускорить работу программного обеспечения [CAD](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F" \o "Система автоматизированного проектирования), использовавшегося при разработке ARM2. Уилсон оптимизировала набор инструкций ARM для исполнения [BBC BASIC](https://ru.wikipedia.org/wiki/BBC_BASIC" \o "BBC BASIC). Изначальная цель компьютера, полностью построенного на базе ARM, была достигнута в 1987 году с выходом Acorn Archimedes.

Атмосфера вокруг проекта ARM была настолько секретна, что когда компания [Olivetti](https://ru.wikipedia.org/wiki/Olivetti" \o "Olivetti) вела переговоры о покупке [контрольного пакета акций](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82_%D0%B0%D0%BA%D1%86%D0%B8%D0%B9" \o "Контрольный пакет акций) Acorn в 1985 году, они не стали рассказывать о развитии проекта до конца переговоров. В 1992 году Acorn ещё раз выиграл Премию королевы для предприятий за ARM.

В ARM2 была 32-разрядная шина данных, 26-битное адресное пространство и 16 32-разрядных регистров. Программный код должен был лежать в первых 64 мегабайтах памяти, а программный счётчик был ограничен 26 битами, так как верхние 4 и нижние 2 бита 32-битного регистра служили флагами. ARM2 стал, возможно, самым простым из популярных 32-битных процессоров в мире, имея всего лишь 30 тысяч транзисторов (для сравнения, в сделанном на 6 лет раньше процессоре [Motorola 68000](https://ru.wikipedia.org/wiki/Motorola_68000" \o "Motorola 68000) было 68 тысяч транзисторов). Многое из этой простоты обусловлено отсутствием [микрокода](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%B4" \o "Микрокод) (который в процессоре 68000 занимает от одной четверти до одной трети площади кристалла) и отсутствием [кэша](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88" \o "Кэш), как и во многих процессорах того времени. Эта простота привела к низким затратам энергии, в то время как ARM был гораздо более производителен, чем [Intel 80286](https://ru.wikipedia.org/wiki/80286" \o "80286)[[9]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-Chattopadhyay2010-9). У его преемника — процессора ARM3 — уже был кэш 4 кб, что ещё больше увеличило производительность.

**Apple, DEC, Intel: ARM6, StrongARM, XScale**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=4" \o "Редактировать раздел \«Apple, DEC, Intel: ARM6, StrongARM, XScale\») | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=4" \o "Редактировать код раздела \«Apple, DEC, Intel: ARM6, StrongARM, XScale\»)]

В конце 1980-х годов [Apple Computer](https://ru.wikipedia.org/wiki/Apple_Computer" \o "Apple Computer) и [VLSI Technology](https://ru.wikipedia.org/w/index.php?title=VLSI_Technology&action=edit&redlink=1" \o "VLSI Technology (страница отсутствует)) начали работать с Acorn Computers над новыми версиями ядра ARM. Работа была настолько важна, что Acorn преобразовала команду разработчиков в 1990 году в новую компанию под названием [Advanced RISC Machines](https://ru.wikipedia.org/wiki/Advanced_RISC_Machines" \o "Advanced RISC Machines). По этой причине ARM иногда расшифровывают *как Advanced RISC Machines* вместо *Acorn RISC Machine*. Advanced RISC Machines стала ARM, когда её родительская компания [ARM Holdings](https://ru.wikipedia.org/wiki/ARM_Holdings" \o "ARM Holdings) вышла на [Лондонскую фондовую биржу](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BD%D0%B4%D0%BE%D0%BD%D1%81%D0%BA%D0%B0%D1%8F_%D1%84%D0%BE%D0%BD%D0%B4%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B1%D0%B8%D1%80%D0%B6%D0%B0" \o "Лондонская фондовая биржа) и [NASDAQ](https://ru.wikipedia.org/wiki/NASDAQ" \o "NASDAQ) в 1998 году.

Новая работа Apple-ARM в конечном итоге превратилась в ARM6, впервые выпущенный в 1992 году. Apple использовала основанный на базе ARM6 процессор ARM610 в качестве основы для своего продукта Apple [Newton](https://ru.wikipedia.org/wiki/Apple_Newton" \o "Apple Newton) PDA. В 1994 году Acorn стала использовать ARM610 как главный процессор в своих компьютерах [RISC PC](https://ru.wikipedia.org/w/index.php?title=RISC_PC&action=edit&redlink=1" \o "RISC PC (страница отсутствует)). Компания [DEC](https://ru.wikipedia.org/wiki/DEC" \o "DEC) также купила лицензию на архитектуру ARM6 (чем вызвала небольшую путаницу, поскольку они также производили процессоры [Alpha](https://ru.wikipedia.org/wiki/DEC_Alpha" \o "DEC Alpha)) и начала производить StrongARM. На 233 МГц этот процессор требовал всего 1 Вт мощности (более поздние версии требовали гораздо меньше). Позднее [Intel](https://ru.wikipedia.org/wiki/Intel" \o "Intel) получил права на эту работу в результате судебного процесса. Intel воспользовалась возможностью дополнить свою устаревшую линейку I960 процессором StrongARM и позднее разработала свою версию ядра под торговой маркой [XScale](https://ru.wikipedia.org/wiki/Intel_XScale" \o "Intel XScale), которую они впоследствии продали компании [Marvell](https://ru.wikipedia.org/wiki/Marvell_Technology_Group" \o "Marvell Technology Group).

Ядро ARM сохранило все тот же размер после всех этих изменений. У ARM2 было 30 тысяч транзисторов.

**Влияние ARM-технологии на рынок**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=5" \o "Редактировать раздел \«Влияние ARM-технологии на рынок\») | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=5" \o "Редактировать код раздела \«Влияние ARM-технологии на рынок\»)]

В основном процессоры семейства завоевали сегмент массовых мобильных продуктов (сотовые телефоны, карманные компьютеры) и встраиваемых систем средней и высокой производительности (от сетевых маршрутизаторов и точек доступа до телевизоров). Отдельные компании заявляют о разработках эффективных серверов на базе кластеров ARM-процессоров[[10]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-10), но пока это только экспериментальные проекты с 32-битной архитектурой[[11]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)" \l "cite_note-11).

Предыдущий лидер списка — суперкомпьютер [Summit (OLCF-4)](https://ru.wikipedia.org/wiki/Summit_(%D1%81%D1%83%D0%BF%D0%B5%D1%80%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)" \o "Summit (суперкомпьютер)), разработанный совместно компаниями [IBM](https://ru.wikipedia.org/wiki/IBM" \o "IBM) (серверные узлы), [Mellanox](https://ru.wikipedia.org/wiki/Mellanox" \o "Mellanox) (межсоединение) и [Nvidia](https://ru.wikipedia.org/wiki/Nvidia" \o "Nvidia) (графические ускорители), и установленный в [Ок-Риджской национальной лаборатории](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA-%D0%A0%D0%B8%D0%B4%D0%B6%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D1%8F" \o "Ок-Риджская национальная лаборатория) — стал вторым, уступив почетное первое место новой японской топ-системе Fugaku ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Русский язык) *[Фугаку](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%B3%D0%B0%D0%BA%D1%83" \o "Фугаку)*), которая показала результат [High Performance/Highly Parallel Linpack (HPL)](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%8B_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8_LINPACK#HPLinpack), равный 415,5 петафлопс. Данный показатель превосходит возможности Summit в 2,8 раза. Fugaku оснащен 48-ядерным процессором A64FX SoC фирмы Fujitsu, таким образом, японская разработка стала первой в истории системой № 1 в списке ТOP500, при этом оснащенной процессорами ARM. При одинарной или более низкой точности, которая часто используется для задач машинного обучения и искусственного интеллекта, пиковая производительность Fugaku составляет более 1000 петафлопс (1 экзафлопс). Новая система установлена в Центре вычислительных наук RIKEN (R-CCS) в Кобе, Япония.

**Процессоры ARM**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=6) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=6)]

*Основная статья:*[***Список архитектур ARM***](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80_ARM)

В настоящее время значимыми являются несколько семейств процессоров ARM:

* [ARM7](https://ru.wikipedia.org/wiki/ARM7) (с тактовой частотой до 60-72 МГц), предназначенные, например, для недорогих мобильных телефонов и встраиваемых решений средней производительности. В настоящее время активно вытесняется новым семейством [Cortex](https://ru.wikipedia.org/wiki/Cortex).
* [ARM9](https://ru.wikipedia.org/wiki/ARM9), [ARM11](https://ru.wikipedia.org/wiki/ARM11) (с частотами до 1 ГГц) для более мощных телефонов, карманных компьютеров и встраиваемых решений высокой производительности.
* Cortex A — новое семейство процессоров на смену [ARM9](https://ru.wikipedia.org/wiki/ARM9) и ARM11.
* [Cortex M](http://www.arm.com/products/processors/cortex-m) — новое семейство процессоров на смену ARM7, также призванное занять новую для ARM нишу встраиваемых решений низкой производительности. В семействе присутствуют четыре значимых ядра:
  + Cortex-M0, Cortex-M0+ (более энергоэффективное) и Cortex-M1 (оптимизировано для применения в ПЛИС) с архитектурой ARMv6-M;
  + Cortex-M3 с архитектурой ARMv7-M;
  + Cortex-M4 (добавлены SIMD-инструкции, опционально FPU) и [Cortex-M7](https://www.arm.com/products/processors/cortex-m/cortex-m7-processor.php) (FPU с поддержкой чисел одинарной и двойной точности) с архитектурой ARMv7E-M;
  + [Cortex-M23](https://www.arm.com/products/processors/cortex-m/cortex-m23-processor.php) и [Cortex-M33](https://www.arm.com/products/processors/cortex-m/cortex-m33-processor.php) с архитектурой [ARMv8-M](https://www.arm.com/products/processors/instruction-set-architectures/armv8-m-architecture.php) [ARMv8-M](https://en.wikipedia.org/wiki/ARM_architecture#ARMv8-R_and_ARMv8-M) .

В 2010 году производитель анонсировал процессоры Cortex-A15 под кодовым названием Eagle, ARM утверждает, что ядро Cortex A15 на 40 процентов производительнее на той же частоте, чем ядро [Cortex-A9](https://ru.wikipedia.org/wiki/ARM_Cortex-A9_MPCore) при одинаковом числе ядер на чипе[[12]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-12). Изделие, изготовленное по 28-нанометровому техпроцессу, имеет 4 ядра, может функционировать на частоте до 2,5 ГГц и будет поддерживаться многими современными операционными системами[[13]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-13)[[14]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-14).

Популярное семейство микропроцессоров [xScale](https://ru.wikipedia.org/wiki/XScale) фирмы Marvell (до 27 июня 2007 года — [Intel](https://ru.wikipedia.org/wiki/Intel)[[15]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-15)) в действительности является расширением архитектуры [ARM9](https://ru.wikipedia.org/wiki/ARM9), дополненной набором инструкций [Wireless MMX](https://ru.wikipedia.org/w/index.php?title=Wireless_MMX&action=edit&redlink=1), специально разработанных фирмой Intel для поддержки мультимедийных приложений.

**Версии ядра ARM (таблица)показать**

**Архитектура**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=7) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=7)]

Уже давно существует справочное руководство по архитектуре ARM, которое разграничивает все типы интерфейсов, которые поддерживает ARM, так как детали реализации каждого типа процессора могут различаться. Архитектура развивалась с течением времени и, начиная с ARMv7, были определены 3 профиля:

* A (application) — для устройств, требующих высокой производительности (смартфоны, планшеты);
* R (real time) — для приложений, работающих в реальном времени;
* M (microcontroller) — для микроконтроллеров и недорогих встраиваемых устройств[[3]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-elec-2010-cortex-3).

Профили могут поддерживать меньшее количество команд (команды определенного типа).

**Режимы**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=8) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=8)]

Процессор может находиться в одном из следующих операционных режимов:

* User mode — обычный режим выполнения программ. В этом режиме выполняется большинство программ.
* Fast Interrupt (FIQ) — режим быстрого прерывания (меньшее время срабатывания).
* Interrupt (IRQ) — основной режим прерывания.
* System mode — защищённый режим для использования операционной системой.
* Abort mode — режим, в который процессор переходит при возникновении ошибки доступа к памяти (доступ к данным или к инструкции на этапе prefetch конвейера).
* Supervisor mode — привилегированный пользовательский режим.
* Undefined mode — режим, в который процессор входит при попытке выполнить неизвестную ему инструкцию.

Переключение режима процессора происходит при возникновении соответствующего исключения или же модификацией регистра статуса.

**Набор команд**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=9) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=9)]

Чтобы сохранить устройство чистым, простым и быстрым, оригинальное изготовление ARM было исполнено без [микрокода](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%B4), как и более простой 8-разрядный процессор 6502, используемый в предыдущих микрокомпьютерах от [Acorn Computers](https://ru.wikipedia.org/wiki/Acorn_Computers).

**Набор команд ARM**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=10) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=10)]

Режим, в котором исполняется 32-битный набор команд.

ARM Base Instruction Set:[[49]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-49)

ADC, ADD, AND, B/BL, BIC, CMN, CMP, EOR, LDM, LDR/LDRB, MLA, MOV, MUL, MVN, ORR, RSB, RSC, SBC, STM, STR/STRB, SUB, SWI, SWP, TEQ, TST

**Набор команд Thumb**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=11) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=11)]

*Основная статья:*[***Thumb***](https://ru.wikipedia.org/wiki/Thumb)

Для улучшения плотности кода процессоры, начиная с ARM7TDMI, снабжены режимом «thumb». В этом режиме процессор выполняет альтернативный набор 16-битных команд. Большинство из этих 16-разрядных команд переводится в нормальные команды ARM. Уменьшение длины команды достигается за счёт сокрытия некоторых операндов и ограничения возможностей адресации по сравнению с режимом полного набора команд ARM.

В режиме Thumb меньшие коды операций обладают меньшей функциональностью. Например, только ветвления могут быть условными, и многие коды операций имеют ограничение в виде доступа только к половине главных регистров процессора. Более короткие коды операций в целом дают большую плотность кода, хотя некоторые операции требуют дополнительных команд. В ситуациях, когда порт памяти или ширина шины ограничены 16 битами, более короткие коды операций режима Thumb становятся гораздо производительнее по сравнению с обычным 32-битным ARM-кодом, так как меньший программный код придется загружать в процессор при ограниченной пропускной способности памяти.

Аппаратные средства типа Game Boy Advance, как правило, имеют небольшой объём оперативной памяти, доступной с полным 32-битным информационным каналом. Но большинство операций выполняется через 16-битный или более узкий информационный канал. В этом случае имеет смысл использовать Thumb-код и вручную оптимизировать некоторые тяжелые участки кода, используя переключение в режим полных 32-битных инструкций ARM.

Первым процессором с декодером Thumb-команд был ARM7TDMI. Все процессоры семейства [ARM9](https://ru.wikipedia.org/wiki/ARM9), а также [XScale](https://ru.wikipedia.org/wiki/Intel_XScale), имели встроенный декодер Thumb-команд.

**Набор команд Thumb-2**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=12) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=12)]

Thumb-2 — технология, появившаяся в ARM1156 core, анонсированном в 2003 году. Он расширяет ограниченный 16-битный набор команд Thumb дополнительными 32-битными командами, чтобы задать набору команд дополнительную ширину. Цель Thumb-2 — достичь плотности кода, как у Thumb, и производительности, как у набора команд ARM на 32 битах. Можно сказать, что в ARMv7 эта цель была достигнута.

Thumb-2 расширяет как команды ARM, так и команды Thumb ещё большим количеством команд, включая управление битовым полем, табличное ветвление, условное исполнение. Новый язык «Unified Assembly Language» (UAL) поддерживает создание команд, как для ARM, так и для Thumb из одного и того же исходного кода. Версии Thumb на ARMv7 выглядят, как код ARM. Это требует осторожности и использования новой команды if-then, которая поддерживает исполнение до 4 последовательных команд испытываемого состояния. Во время компиляции в ARM-код она игнорируется, но во время компиляции в код Thumb-2 генерирует команды. Например:

*; if (r0 == r1)*

CMP r0, r1

ITE EQ *; ARM: no code ... Thumb: IT instruction*

*; then r0 = r2;*

MOVEQ r0, r2 *; ARM: conditional; Thumb: condition via ITE 'T' (then)*

*; else r0 = r3;*

MOVNE r0, r3 *; ARM: conditional; Thumb: condition via ITE 'E' (else)*

*; recall that the Thumb MOV instruction has no bits to encode "EQ" or "NE"*

Все кристаллы ARMv7 поддерживают набор команд Thumb-2, а некоторые кристаллы, вроде Cortex-m3, поддерживают только Thumb-2. Остальные кристаллы Cortex и ARM11 поддерживают наборы команд как Thumb-2, так и ARM.

**Набор команд Jazelle**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=13) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=13)]

[Jazelle](https://ru.wikipedia.org/wiki/Jazelle) — это технология, которая позволяет [байткоду](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82%D0%BA%D0%BE%D0%B4) [Java](https://ru.wikipedia.org/wiki/Java) исполняться прямо в архитектуре ARM в качестве 3-го состояния исполнения (и набора команд) наряду с обычными командами ARM и режимом Thumb. Поддержка технологии Jazelle обозначается буквой «J» в названии процессора — например, ARMv5TEJ. Данная технология поддерживается начиная с архитектуры ARMv6, хотя новые ядра содержат лишь ограниченные реализации, которые не поддерживают аппаратного ускорения.

**ARMv8 и набор команд ARM 64 бита**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=14) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=14)]

В конце 2011 года была опубликована новая версия архитектуры, ARMv8. В ней появилось определение архитектуры AArch64, в которой исполняется 64-битный набор команд A64. Поддержка 32-битных команд получила название A32 и исполняется на архитектурах AArch32. Инструкции Thumb поддерживаются в режиме T32, только при использовании 32-битных архитектур. Допускается исполнение 32-битных приложений в 64-битной ОС, и запуск виртуализованной 32-битной ОС при помощи 64-битного гипервизора[[50]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-50). Applied Micro, AMD, Broadcom, Calxeda, HiSilicon, Samsung, STM и другие заявили о планах по использованию ARMv8. Ядра Cortex-A53 и Cortex-A57, поддерживающие ARMv8, были представлены компанией ARM 30 октября 2012 года[[51]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-51).

Как AArch32, так и AArch64 поддерживают VFPv3, VFPv4 и *advanced SIMD* (NEON). Также добавлены криптографические инструкции для работы с [AES](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard), [SHA-1](https://ru.wikipedia.org/wiki/SHA-1) и [SHA-256](https://ru.wikipedia.org/wiki/SHA-2).

Особенности AArch64:

* Новый набор команд A64
* 31 регистр общего назначения, каждый длиной 64 бита
* Отдельные регистры SP и PC
* Инструкции имеют размер 32 бита и многие совпадают с командами A32
* Большинство инструкций работает как с 32-, так и с 64-битными аргументами
* Адреса имеют размер 64 бита
* Улучшения Advanced SIMD (NEON) enhanced
* С 16 до 32 увеличено количество 128-битных регистров, доступных через NEON, VFPv4, криптоинструкции AES, SHA
* Поддерживает вычисления с числами с плавающей запятой двойной точности (64-бит double)
* Полная совместимость с IEEE 754
* Новая система исключений
* Трансляция виртуальных адресов из 48-битного формата работает с помощью существующих механизмов LPAE

**Функции RISC**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=15) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=15)]

Архитектура ARM обладает следующими особенностями RISC:

* Архитектура загрузки/хранения
* Нет поддержки нелинейного (не выровненного по словам) доступа к памяти (теперь поддерживается в процессорах ARMv6, за некоторыми исключениями, и полностью в ARMv7)
* Равномерный 16х32-битный регистровый файл
* Фиксированная длина команд (32 бита) для упрощения декодирования за счет снижения плотности кода. Позднее режим Thumb повысил плотность кода.
* Однотактное исполнение

Чтобы компенсировать простой дизайн, в сравнении с современными процессорами вроде Intel 80286 или Motorola 68020 были использованы некоторые особенности дизайна:

* Арифметические инструкции заменяют условные коды, только когда это необходимо
* 32-битное многорегистровое циклическое сдвиговое устройство, которое может быть использовано без потерь производительности в большинстве арифметических инструкций и адресных расчетов.
* Мощные индексированные адресные режимы
* Регистр ссылок для быстрого вызова функций листьев
* Простые, но быстрые, с двумя уровнями приоритетов подсистемы прерываний с включенными банками регистров.

**Условное исполнение**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=16) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=16)]

Одним из существенных отличий архитектуры ARM (изначальная архитектура) от других архитектур ЦПУ является так называемая *предикация* — возможность условного исполнения команд. Под «условным исполнением» здесь понимается то, что команда будет выполнена или проигнорирована в зависимости от текущего состояния флагов состояния процессора. В Thumb и Arm 64 предикация не используется — в первом режиме для неё нет места в команде (всего 16 бит), а во втором — предикация бессмысленна[[*источник не указан 3741 день*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)] и сложна для реализации на [суперскалярных архитектурах](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

В то время как для других архитектур таким свойством, как правило, обладают только команды условных переходов, в архитектуру ARM была заложена возможность условного исполнения практически любой команды. Это было достигнуто добавлением в коды их инструкций особого 4-битового поля (**предиката**). Одно из его значений зарезервировано на то, что инструкция должна быть выполнена безусловно, а остальные кодируют то или иное сочетание условий (флагов). С одной стороны, с учётом ограниченности общей длины инструкции, это сократило число битов, доступных для кодирования смещения в командах обращения к памяти, но с другой — позволило избавляться от инструкций ветвления при генерации кода для небольших if-блоков.

Пример, обычно рассматриваемый для иллюстрации — основанный на вычитании [алгоритм Евклида](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0). В языке C он выглядит так:

**while** (i != j) {

**if** (i > j) {

i -= j;

} **else** {

j -= i;

}

}

А на ассемблере ARM — так:

loop CMP Ri, Rj *; set condition «NE» if (i != j),*

*; "GT" if (i > j),*

*; or "LT" if (i < j)*

SUBGT Ri, Ri, Rj *; if "GT" (greater than), i = i-j;*

SUBLT Rj, Rj, Ri *; if "LT" (less than), j = j-i;*

BNE loop *; if "NE" (not equal), then loop*

Из кода видно, что использование предикации позволило полностью избежать ветвления в операторах else и then. Заметим, что если Ri и Rj равны, то ни одна из SUB-инструкций не будет выполнена, полностью убирая необходимость в ветке, реализующей проверку while при каждом начале цикла, что могло быть реализовано, например, при помощи инструкции SUBLE (меньше либо равно).

Один из способов, которым уплотнённый (Thumb) код достигает большей экономии объёма — это именно удаление 4-битового предиката из всех инструкций, кроме ветвлений.

**Другие особенности**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=17) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=17)]

Другая особенность набора команд — это возможность соединять сдвиги и вращения в инструкции «обработки информации» (арифметическую, логическую, движение регистр-регистр) так, что, например, выражение С:

a += (j << 2);

может быть преобразовано в команду из одного слова и одного цикла в ARM:

ADD Ra, Ra, Rj, LSL #2

Это приводит к тому, что типичные программы ARM становятся плотнее, чем обычно, с меньшим доступом к памяти. Таким образом, конвейер используется гораздо более эффективно. Даже несмотря на то, что ARM работает на скоростях, которые многие бы сочли низкими, он довольно-таки легко конкурирует с многими более сложными архитектурами ЦПУ.

ARM-процессор также имеет некоторые особенности, редко встречающиеся в других архитектурах RISC — такие, как адресация относительно счетчика команд (на самом деле счетчик команд ARM является одним из 16 регистров), а также пре- и постинкрементные режимы адресации.

Другая особенность, которую стоит отметить, — это то, что некоторые ранние ARM-процессоры (до ARM7TDMI), например, не имеют команд для хранения 2-байтных чисел. Таким образом, строго говоря, для них невозможно сгенерировать эффективный код, который бы вел себя так, как ожидается от объектов С типа «volatile int16\_t».

**Конвейер и другие аспекты реализации**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=18) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=18)]

ARM7 и более ранние версии имеют трехступенчатый конвейер. Это ступени переноса, декодирования и исполнения. Более производительные архитектуры, типа [ARM9](https://ru.wikipedia.org/wiki/ARM9), имеют более сложные конвейеры. Cortex-a8 имеет 13-ступенчатый конвейер.

**Сопроцессоры**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=19) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=19)]

Архитектура предоставляет способ расширения набора команд, используя сопроцессоры, которые могут быть адресованы, используя MCR, MRC, MRRC, MCRR и похожие команды. Пространство сопроцессора логически разбито на 16 сопроцессоров с номерами от 0 до 15, причем 15-й зарезервирован для некоторых типичных функций управления, типа управления кэш-памятью и операции [блока управления памятью](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%D1%8E) (на процессорах, в которых они есть).

В машинах на основе ARM периферийные устройства обычно подсоединяются к процессору путём сопоставления их физических регистров в памяти ARM или в памяти сопроцессора, или путём присоединения к шинам, которые, в свою очередь, подсоединяются к процессору. Доступ к сопроцессорам имеет большее время ожидания, поэтому некоторые периферийные устройства проектируются для доступа в обоих направлениях. В остальных случаях разработчики чипов лишь пользуются механизмом интеграции сопроцессора. Например, движок обработки изображений должен состоять из малого ядра ARM7TDMI, совмещенного с сопроцессором, который поддерживает примитивные операции по обработке элементарных кодировок HDTV.

**Усовершенствованный SIMD (NEON)**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=20) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=20)]

Расширение усовершенствованного SIMD, также называемое технологией NEON — это комбинированный 64- и 128-битный набор команд [SIMD](https://ru.wikipedia.org/wiki/SIMD) (single instruction multiple data), который обеспечивает стандартизованное ускорение для медиаприложений и приложений обработки сигнала. NEON может выполнять декодирование аудиоформата [mp3](https://ru.wikipedia.org/wiki/Mp3) на частоте процессора в 10 МГц, и может работать с речевым кодеком [GSM](https://ru.wikipedia.org/wiki/GSM) [AMR](https://ru.wikipedia.org/wiki/AMR_(%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B7%D0%B2%D1%83%D0%BA%D0%B0)) (adaptive multi-rate) на частоте не более 13 МГц. Он обладает внушительным набором команд, отдельными регистровыми файлами, и независимой системой исполнения на аппаратном уровне. NEON поддерживает 8-, 16-, 32-, 64-битную информацию целого типа, одинарной точности и с плавающей запятой, и работает в операциях SIMD по обработке аудио и видео (графика и игры). В NEON SIMD поддерживает до 16 операций единовременно.

Одним из недостатков (или, скажем, особенностью) усовершенствованного SIMD является то, что сопроцессор выполняет команды усовершенствованного SIMD с достаточно значительной задержкой относительно кода основного процессора, задержка достигает двух десятков тактов и более (зависит от архитектуры и конкретных условий). По этой причине при попытке основного процессора воспользоваться результатами вычисления сопроцессора исполнение будет заморожено на значительное время.

**VFP**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=21) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=21)]

Технология VFP (Vector Floating Point, вектора чисел с плавающей запятой) — расширение сопроцессора в архитектуре ARM. Она производит низкозатратные вычисления над числами с плавающей запятой одинарной/двойной точности, в полной мере соответствующие стандарту [ANSI/IEEE Std 754—1985 Standard for Binary Floating-Point Arithmetic](https://ru.wikipedia.org/wiki/IEEE_754). VFP производит вычисления с плавающей запятой, подходящие для широкого спектра приложений — например, для КПК, смартфонов, сжатие звука, трёхмерной графики и цифрового звука, а также принтеров и телеприставок. Архитектура VFP также поддерживает исполнение коротких векторных команд. Но, поскольку процессор выполняет операции последовательно над каждым элементом вектора, то VFP нельзя назвать истинным SIMD-набором инструкций. Этот режим может быть полезен в графике и приложениях обработки сигнала, так как он позволяет уменьшить размер кода и выработку команд.

Другие сопроцессоры с плавающей запятой и/или SIMD, находящиеся в ARM-процессорах, включают в себя [FPA](https://ru.wikipedia.org/w/index.php?title=Floating_Point_Accelerator&action=edit&redlink=1), [FPE](https://ru.wikipedia.org/wiki/FPE), [iwMMXt](https://ru.wikipedia.org/w/index.php?title=IwMMXt&action=edit&redlink=1). Они обеспечивают ту же функциональность, что и VFP, но не совместимы с ним на уровне [опкодов](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%BA%D0%BE%D0%B4).

**Расширения безопасности**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=22) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=22)]

Расширения безопасности, позиционируемые как TrustZone Technology, находятся в ARMv6KZ и других, более поздних, профилированных на приложениях архитектурах. Оно обеспечивает низкозатратную альтернативу добавлению специального ядра безопасности, обеспечивая 2 виртуальных процессора, поддерживаемых аппаратным контролем доступа. Это позволяет ядру приложения переключаться между двумя состояниями, называемыми «миры» (чтобы избежать путаницы с названиями возможных доменов), чтобы не допустить утечку информации из более важного мира в менее важный. Этот переключатель миров обычно ортогонален всем другим возможностям процессора. Таким образом, каждый мир может работать независимо от других миров, используя одно и то же ядро. Память и периферия соответственно изготавливаются с учетом особенностей мира ядра, и могут использовать это, чтобы получить контроль доступа к секретам и кодам ядра. Типичные приложения TrustZone Technology должны запускать полноценную операционную систему в менее важном мире, и компактный, специализированный на безопасности, код в более важном мире, позволяя Digital Rights Management’у намного точнее контролировать использование медиа на устройствах на базе ARM, и предотвращая несанкционированный доступ к устройству.

На практике же, так как конкретные детали реализации TrustZone остаются собственностью компании и не разглашаются, остается неясным, какой уровень безопасности гарантируется для этой модели угроз.

**Отладка**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=23) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=23)]

Все современные процессоры ARM имеют аппаратные средства отладки, так как без них отладчики ПО не смогли бы выполнить самые базовые операции типа остановки, отступа, установки контрольных точек после перезагрузки.

Архитектура ARMv7 определяет базовые средства отладки на архитектурном уровне. К ним относятся точки останова, точки просмотра и выполнение команд в режиме отладки. Такие средства были также доступны с модулем отладки EmbeddedICE. Поддерживаются оба режима — остановки и обзора. Реальный транспортный механизм, который используется для доступа к средствам отладки, не специфицирован архитектурно, но реализация, как правило, включает поддержку [JTAG](https://ru.wikipedia.org/wiki/JTAG).

Существует отдельная архитектура отладки «с обзором ядра», которая не требуется архитектурно процессорами ARMv7.

**Регистры**

[[править](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&veaction=edit&section=24) | [править код](https://ru.wikipedia.org/w/index.php?title=ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)&action=edit&section=24)]

ARM предоставляет 31 [регистр общего назначения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D1%8B_%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE_%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) разрядностью 32 бита. В зависимости от режима и состояния процессора пользователь имеет доступ только к строго определённому набору регистров. В ARM state разработчику постоянно доступны 17 регистров:

* 13 регистров общего назначения (r0..r12).
* Stack Pointer (r13) — содержит указатель стека выполняемой программы.
* Link register (r14) — содержит адрес возврата в инструкциях ветвления.
* Program Counter (r15) — биты [31:1] содержат адрес выполняемой инструкции.
* Current Program Status Register (CPSR) — содержит флаги, описывающие текущее состояние процессора. Модифицируется при выполнении многих инструкций: логических, арифметических, и др.

Во всех режимах, кроме User mode и System mode, доступен также Saved Program Status Register (SPSR). После возникновения исключения регистр CPSR сохраняется в SPSR. Тем самым фиксируется состояние процессора (режим, состояние; флаги арифметических, логических операций, разрешения прерываний) на момент непосредственно перед прерыванием[[52]](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)#cite_note-52).