# Компьютеры с сокращенным набором команд

Компьютеры с сокращенным набором команд (КСНК) – более известные как RISC-компьютеры (RISC – Reduced Instruction Set Computer) – воплощают направление развития архитектуры ВМ, связанное с возвращением к принципам аппаратного управления выполнением команд с целью повышения производительности.

Системы команд первого и второго поколений машин содержали не более пятидесяти команд. Причины этого: 1) высокая цена аппаратуры (и управляющая, и обрабатывающая части процессора реализовывались аппаратно), 2) необходимость программирования в кодах, а программист не мог запомнить большое количество разных команд. Период доминирования аппаратного управления в 50-е – начало 60-х годов, можно назвать **«эрой аппаратчиков»**.

С середины 60-х до 80-х годов стало доминировать микропрограммное управление выполнением команд, воплощающее **«эру программистов»**, основным лозунгом которой было: «Больше команд хороших и разных!».

Этот лозунг соответствовал требованиям к процессорам того времени:

1. Минимизация длины кода программы
2. Упрощение реализации компиляторов за счет снижения семантического разрыва между ЯВУ и машинными командами.

Это вызвало рост набора команд компьютеров за счет увеличения их сложности и увеличения числа форматов: от ~50 до ~300 команд (рекордсменом был компьютер фирмы DEC Vaх11/780, у него было 303 команды). Компьютеры с большим набором команд и разнообразием их форматов получили название CISC-компьютеров (Complex Instruction Set Computer – машина со сложным набором команд). Для них характерно увеличение сложности и соответственно размеров микропрограммного устройства управления, которое интерпретирует выполнение этих команд.

Так в процессорах iX86 в мат. сопроцессоре используются команды: F2XM1 – реализует f = 2х -1 или FYL2XP1 - реализует f = y\*log2(x+1) .

В это время тактовая частота процессоров стала достигать 100 Мгц и повышение производительности требовало размещения всех частей процессора на одном кристалле для сокращения длины соединений его элементов. Но микропрограммное управление CISC из-за своей сложности занимало до 60% площади кристалла, что либо не допускало эффективных средств арифметической обработки данных, либо требовало размещения частей процессора на разных кристаллах.

Все это приводило к существенному ограничению производительности, увеличивало сроки разработки и снижало выход годных кристаллов.

В 80-х годах рядом исследователей было замечено, что при выполнении большинства программ наиболее активно используется только около 30% сравнительно простых команд арифметики и управления. Вследствие этого стало формироваться направление развития архитектуры компьютеров, требующее чтобы система команд процессора содержала минимальный набор наиболее часто используемых и наиболее простых команд (возврат к примерно 50 командам). Это направление получает название компьютеров с сокращенным набором команд или **RISC-компьютеров**  и имеет лозунг: «Меньше команд и их упрощение - выше скорость выполнения!».

В результате в конце 80-х годов постепенно сложились основные принципы (или законы) RISC-архитектур:

1. Основной набор команд не должен интерпретироваться микрокомандами, а должен выполняться аппаратным обеспечением.
2. Все команды должны иметь одинаковую длину и минимальное число форматов (обычно не более 2–3), это упрощает логику управления при выборе и при исполнении команды.
3. Любая команда основного набора должна выполняться за один машинный цикл, обратно пропорциональный тактовой частоте процессора (стандартом является команда сложения регистра с регистром, занимающая от 3–10 нс); это достигается одновременным (параллельным) выполнением максимально возможного числа команд путем конвейеризации или использованием нескольких обрабатывающих узлов.
4. Обращение к памяти производится только по специально выделенным командам работы с памятью типа: Load – загрузка и Store – сохранение, выполняющим групповой обмен регистров с памятью, а ***вся обработка данных должна вестись в регистровом формате***; при этом количество регистров должно быть велико (более 100).
5. Система команд должна обеспечивать поддержку компиляции с конкретного языка программирования (компиляторы для RISC существенно сложнее, чем компиляторы для CISC).

Название RISC появилось в середине 80-х годов в университете Беркли, где под руководством Дэвида Паттерсона и Карло Секвина была создана машина RISC–1, а затем последовало создание RISC–2, позже принятой за основу машин семейства SPARC фирмы Sun. Почти одновременно в Стэнфордском университете был разработан процессор MIPS, положивший начало выпуску семейства машин R4000 – R10000 фирмы MIPS.

Затем почти все ведущие производители ЭВМ стали разрабатывать и выпускать машины на основе RISC-архитектур:

1. Hewlett - Packard – PA7xxx – PA9xxx (PA – Precision Architecture)
2. DEC – Alpha 21xxx : 3) IBM + Apple – Power PC .

*Формат команд и структура процессора RISC–1*

Как и в большинстве RISC-процессоров команды являются трехадресными

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | 1 | 5 | 5 | 1 | 13 |
| КОП | Усл | Dest | SRC1 | IMM | SRC2 |
| 31 25 | 24 | 23 19 | 18 14 | 13 | 12 0 |

КОП – код операции; Усл – бит условия (для команд переходов);

Dest – номер регистра назначения (длина пять бит – NPOH = 32);

SRC1 – номер регистра-источника 1;

SRC2 – регистр (IMM = 0) или непосредственное значение источника 2.

В процессоре RISC-1 используется два вида формата команды.

1. Rd ⇐ Rs1 oper S2 – для выполнения операций обработки;
2. Rd ⇔ M ((Rs1) + S2) – для выполнения обмена с памятью.

Структура процессора RISC -1 показана на рис. 1.

Шина процессор-память

Кэш команд

АЛУ

Буфер предвыборки

и логика управления Блок Блок

Рг1 Рг2

Логика Регистр Логика

выборки резуль- выборки

Рг1 тата Рг2

*Рис. 1*

Регистровый формат команд обработки данных облегчает реализацию конвейерного исполнения команд, так как исключает конфликты при обращении к памяти.

*Пример*. Пусть требуется вычислить выражение E = (A + B) \* (C + D)

1. Двухадресная обработка команд в CISC-компьютерах с использова-нием ячеек памяти приводит к большому числу пропусков тактов в конвейере (такты ожидания) из-за конфликтов за ресурс памяти.

Add b, a

║ ВК | ДШ | ВО | ИСП | ЗР ║ | | | |

║ ВК | ДШ | ВО | ИСП | ЗР ║ | | |

║ …… | …… | …… | …… | ВК | ДШ | ВО | ИСП | ЗР ║ |

║ …… | …… | …… | …… | ВК | ДШ | ВО | ИСП | ЗР |

Add d, c

Mul d, b

Store e, d

ВК – выборка команды, ДШ – дешифрация, ВО – выборка операндов,

ИСП – исполнение операции, ЗР – запись результата, **…** – такт ожидания.

1. Трехадресная обработка команд в RISC-компьютерах на основе регистров практически исключает конфликты при обращении к памяти.

Add R1, R2, R6

║ ВК | ДШ | ИСП ║

║ ВК | ДШ | ИСП ║

║ ВК | ДШ | ИСП ║

Add R3, R4, R7

Mul R6, R7, R5

*Арифметические особенности и производительность RISC-процессоров*

В RISC-процессорах увеличивается аппаратная поддержка арифмети-ческих операций благодаря уменьшению места на кристалле для размещения управляющей части процессора. В АЛУ процессора широко применяются аппаратные методы ускорения выполнением операций (одновременная обработка нескольких разрядов в одном такте, групповые переносы и т.д.).

Время выполнения программы упрощенно можно оценить по выражению:

Tпрограммы = Nкоманд \* Cтакт/ком \* Tтакт ,

где Tтакт – длительность такта, и для параметров CISC- и RISC-процессоров справедливы соотношения:

NкомRISC = (2 - 3) \* NкомCISC

Cтакт/ком.CISC= (10 - 20) \* Cтакт/ком.RISC

TтактCISC = (2 - 3) \* TтактRISC

Тогда для времен выполнения программ с большим числом арифметических операций будет справедливо соотношение:

Тпрогр.CISC = (10 - 20)Тпрогр.RISC.

*Использование «Окон перекрытия регистров».*

Основная проблема RISC-компьютеров: при использовании большого количества регистров для обработки данных в процессоре надо сохранять их в памяти при переключениях с одной процедуры на другую. Решение проблемы сохранения и восстановления набора регистров при вызове подпрограмм в компьютере RISC-1 реализуется путем организации так называемых “Окон перекрытия регистров” – MORS – Multiple Overlapping Register Set (Rolodex в RISC–2) .

Сто тридцать восемь регистров в RISC-1 распределялись между 8 программами (по 32 регистра) как показано на рис. 2.

Физические регистры Л о г и ч е с к и е р е г и с т р ы

Проц. А

R137 R31A

HIGH A

R132 R26A

R131 R25A

Окно А LOCAL A

R122 R16A Проц. B

R121 R15A R31B

LOW A / HIGH B

R116 R10A R26B

R115 R25B

Окно В LOCAL B

R106 R16B Проц. С

R105 R15B R31C

LOW B / HIGH C

R100 R10B R26C

R99 R25C

Окно С LOCAL C

R90 R16C

R89 R15C

LOW C

R84 R10C

**. . .**

R9

Global R

R0

Nрег = 10 + 16\*8 = 138

*Рис. 2*

Передача параметров (до 6 штук) между процедурами реализуется без затрат времени просто за счет перекрытия логических регистров (физические регистры совпадают).

Недостатки MORS:

1. Ограниченное число параметров при передаче через регистры.
2. Ограниченное число подпрограмм, которым могут предоставляться регистровые окна.

*Достоинства и недостатки RISC- процессоров*

Основные ***достоинства RISC- процессоров:***

1. Повышение производительности обработки программ вычислитель-ных задач.
2. Благодаря использованию простых команд и минимума их форматов сокращается время разработки RISC-процессора.
3. Улучшение технологичности RISC-процессоров благодаря большей свободе в размещении их элементов на кристалле интегральной схемы и повышение вероятности выхода годных схем.

***Недостатки RISC- процессоров:***

1. Нарушение основных принципов программирования:

а) Минимум длины исполняемого кода программы.

б) Снижение семантического разрыва между исходным описанием программы и машинным кодом.

1. Усложнение построения компилятора, поскольку программа с языка высокого уровня должна транслироваться в микрокод с оптимизацией использования регистров.
2. Высокие требования к быстродействию памяти.

Современные процессоры, как правило, реализуются *по гибридному принципу:* содержат ***ядро RISC***, которое выполняет простые и самые распространенные команды за 1 такт, а сложные команды выполняются как последовательность микрокоманд.

Cледует учитывать, что для многих задач производительность вычисли-тельной системы определяется ***не процессором***, а скоростью передачи данных по шинам и протоколами взаимодействия. Также важным требованием является совместимость с созданным ранее ПО, которому многие RISC-процессоры не удовлетворяют, что сдерживает их распространение.

***Развитие архитектур процессоров на основе концепции RISC.***

Преимущества концепции RISC активно стали использовать в британской компании ARM Limited. Они внесли различные усовершенствования в эту концепцию и уже в 1985 году представили свой первый процессор ARM1. В последствие под ARM (Advanced RISC Machine) начали понимать авторскую лицензируемую архитектуру RISC-процессоров .

Поскольку изначально ARM задумывалась как усовершенствование архитектуры RISC, то для ранних версий процессоров были характерны некоторые общие черты:

* фиксированный размер команд;
* большое число регистров общего назначения;
* специализированные команды для операций с памятью — чтения или записи.

Однако постепенно разработчики ARM вносили все больше изменений и модификаций, отклоняясь от классической концепции RISC. Например, из 31 регистра программисту доступно всего 16, что совсем мало. Также в некоторых случаях одна команда в ARM может выполнять не только обработку данных (например, сложение) но и сдвиг. Это позволяет частично уменьшить сложность кода, но за счет увеличения сложности самих команд. Еще одно важная особенность ARM — это развитый набор видов адресации, который не предусмотрен в концепции RISC архитектуры. Помимо этого в ARM применяется условное исполнение, когда каждая команда может исполняться или не исполняться в зависимости от определенных предуста-новок (установленных флагов).

В результате получается, что архитектура ARM — это более дешевый и унифицированный, но из-за этого менее универсальный вариант, чем процессоры на x86-архитектуре. Ориентация на концепцию RISC сделали ARM отличным вариантом для процессоров мобильной электроники. Однако в них граница между архитектурами х86 и RISC становится все менее явной.

Во-первых, в x86-процессорах уже используется разбиение длинных инструкций на более мелкие. Фактически, в CISC-процессорах применяется RISC-ядро, что позволяет повысить производительность. Во-вторых, функциональность ARM процессоров постепенно приближается к x86. Соответственно, процессоры на этой архитектуре, как и x86, уже могут использоваться в ноутбуках и даже ПК. Ключевое отличие этих двух архитектур заключается в системе команд, что свою очередь приводит к различным конструктивным различиям, вроде числа регистров и дополнительных модулей.

Переходы в х86 на новые техпроцессы и многоядерность позволяют успешно использовать х86 для высокопроизводительной техники вроде ноутбуков, персональных компьютеров и серверов.

В свою очередь ARM хоть и задумывалась изначально как RISC-подобная, со временем обросла множеством усовершенствований. Это дает приближен-ную к десктопным процессорам эффективность, но приводит к большему размеру кристалла и тепловыделению.

**Архитектура RISC-V**

Итак, ARM и x86 — это своеобразные противоположности, у каждой из которых свои достоинства и недостатки. Архитектура x86 не подходит для мобильной электроники, но ARM требует лицензирования. Если вы захотите спроектировать свой процессор на архитектуре ARM, то придется заплатить не менее одного миллиона долларов.

Над этим вопросом задумались в отделении информатики Калифорнийского университета в Беркли и в 2010 году группа инженеров под руководством Крсте Асановича и Дэвида Паттерсона представили процессор RISC-V, главная отличительная черта которого — полностью открытая архитектура, которую можно использовать абсолютно бесплатно. Кроме того, RISC-V должна предусматривать единую систему команд для всех типов ЭВМ от микроконтроллеров до высокопроизводительных машин.

То есть RISC-V — это улучшенная версия классической архитектуры RISC, созданной в Беркли несколько десятилетий назад. Главное достоинство RISC-V в том, что ее архитектура подходит для самых разнообразных вычислительных задач. Базовая версия RISC-V содержит минимальный набор инструкций — всего 47, но при необходимости его можно расширять для решения своей задачи. То есть RISC-V - это модульная структура или базовый конструктор, на основе которого можно проектировать собственные системы.

В результате RISC-V позволяет достигать целый ряд практических целей:

1. Повышение энергоэффективности и производительности, уменьшение числа необходимых транзисторов, удешевление разработки и поддержки процессорных ядер - в минимальном варианте разработчики обязаны реализовывать лишь относительно небольшой набор инструкций для получения полноценно работающей системы. После чего они могут сосредоточиться на интересующих именно их областях применения.

2. Модульность и расширяемость. Минимальные наборы команд существуют и в других процессорах. Основная проблема существующих систем команд в том, что небольшой набор команд обычно не универсален. Он бывает ориентирован для областей, требующих специфической аппаратной функциональности (высокопроизводительные вычисления, многопоточность или графика). RISC-V решает эту проблему благодаря модульной структуре системы команд. Разработчики ЭВМ могут реализовывать существующие расширения (например, RVF32 для работы с плавающей точкой), либо добавлять свои собственные – система команд спроектирована с прицелом на расширение. Это в корне отличает RISC-V, например, от ARM, где не только расширяться проблематично, но и сама компания ARM запрещает подобные расширения по условиям лицензии.

3. Удешевление разработки различных систем за счёт создания единой открытой экосистемы, включающей: компиляторы, операционные системы, драйверы, периферию. Наличие **открытых** репозиториев, в которых многое уже реализовано и за счёт многолетней поддержки сообщества доведено до высокого качества, упрощает разработку новых систем.

4. Обратная совместимость программного обеспечения – программы, разработанные для старых процессоров (вернее, процессоров с поддержкой меньшего количества расширений), должны работать на новых (вернее, процессорах той же системы команд, но с большим количеством реализованных расширений) без перекомпиляции. В случае зоопарка коммерческих систем команд вроде x64 или ARM это сделать существенно труднее.

5. Безопасность. Если в некотором RISC-V процессоре обнаружена уязвимость, вы можете легко заменить его на другой RISC-V процессор, пусть и не такой эффективный, но зато без закладок потенциального противника. При этом вы сможете запустить на нём всё существующее ПО с минимальными затратами усилий и времени.

RISC-V и ARM-архитектуры основаны на одном подходе — RISC. В связи с этим они обе подходят для микроэлектроники. Однако ARM — платная и по ходу своего развития претерпела множество изменений, создав целую линейку готовых ядер для разных нужд.

В 2015 г. был основан Фонд RISC-V для продвижения новой архитектуры среди коммерческих пользователей. Проект стремительно завоевал популярность у исследователей и ИТ-компаний по всему миру, и сейчас эта некоммерческая организация насчитывает более двух тысяч членов и партнеров в лице Google, IBM, NVIDIA, Western Digital и других крупных компаний. В 2018 г. организация объявила о сотрудничестве с Linux Foundation, а в 2022 г. компания  [Intel заявила о присоединении к RISC-V International и инвестировании одного миллиарда долл. в экосистему RISC-V](https://www.networkworld.com/article/3649114/intel-partners-with-risc-v-invests-1b-in-foundry-ecosystem.html" \t "_blank).

В РФ одним из лидеров экосистемы RISC-V и основным разработчиком микропроцессорных ядер на основе RISC-V является петербургская компания Syntacore, представившая открытое ядро SCR1 в 2019 г. В настоящее время партнер Ростеха компания Yadro выкупила 51-процентную долю у компании [Syntacore](https://syntacore.com/" \t "_blank) и является инициатором разработки МП-ядер на основе RISC-V. Ориентация на RISC-V дает российским разработчикам аппаратного обеспечения возможность снизить зависимость от зарубежных поставщиков и создавать технологические решения в соответствии со стратегией импортозамещения.