0
Рейтинг

Droider.Ru

istishev
18 сен 2020 в 16:29

ARM против x86: В чем разница между двумя архитектурами процессоров?

6 мин 177K

Блог компании Droider.Ru, Программирование*, Смартфоны, Ноутбуки, Процессоры

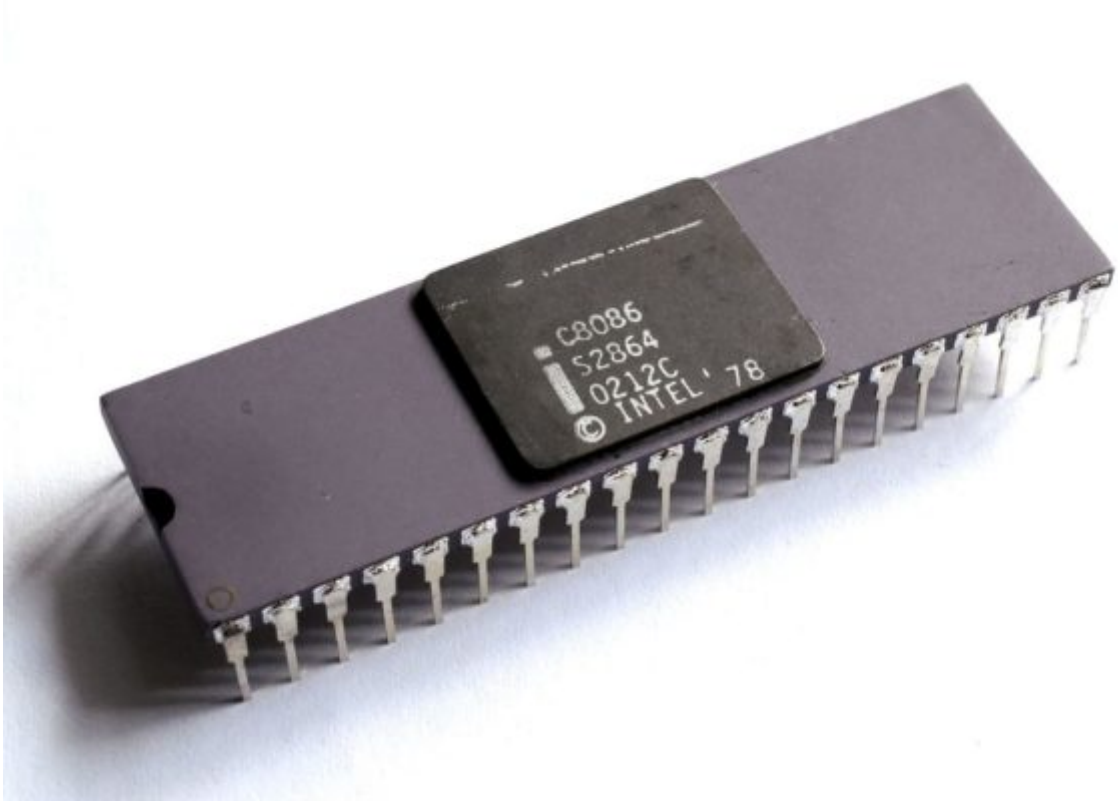
Вы наверняка знаете, что мир процессоров разбит на два лагеря. Если вы смотрите это видео со смартфона, то для вас работает процессор на архитектуре ARM, а если с ноутбука, для вас трудится чип на архитектуре x86.

А теперь еще и Apple объявила, что переводит свои Mac на собственные процессоры Apple Silicon на архитектуре ARM. Мы уже рассказывали, почему так происходит. А сегодня давайте подробно разберемся, в чем принципиальные отличия x86 и ARM. И зачем Apple в это все вписалась?

Итак, большинство мобильных устройств, iPhone и Android'ы работают на ARM'е. Qualcomm, HUAWEI Kirin, Samsung Exynos и Apple A13/A14 Bionic — это все ARM-процессоры.

А вот на компьютере не так — там доминирует x86 под крылом Intel и AMD. Именно поэтому на телефоне мы не можем запустить Word с компьютера.

x86 — так называется по последним цифрам семейства классических процессоров Intel 70-80х годов.



Чем же они отличаются?

Есть два ключевых отличия.

Первое — это набор инструкций, то есть язык который понимает процессор

x86 процессоры используют сложный набор инструкций, который называется CISC - Complex Instruction Set Computing.

ARM процессоры наоборот используют упрощенный набор инструкций — RISC - Reduced Instruction Set Computing.

Кстати ARM расшифровывается как Продвинутые RISC машины - Advanced RISC Machines.

Наборы инструкций ещё принято назвать архитектурой или ISA - Instruction Set Architecture.

Второе отличие — это микроархитектура. Что это такое?

От того на каком языке говорят процессоры, зависит и то, как они проектируются. Потому как для выполнения каждой инструкции на процессоре нужно расположить свой логический блок. Соответственно, разные инструкции — разный дизайн процессора. А дизайн — это и есть микроархитектура.

- x86 — CISC
- ARM — RISC

Итак, запомнили. Говорим x86 — подразумеваем архитектуру CISC, ARM — это RISC.

Но как так произошло, что процессоры стали говорить на разных языках?

История CISC

Памятка программиста, 1960-е годы. Цифровой (машинный) код «Минск-22».

Всё началось в 1960-х. Поначалу программисты работали с машинным кодом, то есть реально писали нолики и единички. Это быстро всех достало и появился Assembler. Низкоуровневый язык программирования, который позволял писать простые команды типа сложить, скопировать и прочее. Но программировать на Assembler'е тоже было несладко. Потому как приходилось буквально “за ручку” поэтапно описывать процессору каждое его действие.

Поэтому, если бы вы ужинали с процессором, и попросили передать его вам соль, это выглядело бы так:

- Эй процессор, посмотри в центр стола.
- Видишь соль? Возьми её.

- Теперь посмотри на меня.
- Отдай мне соль. — Ага, спасибо!
- А теперь снова возьми у меня соль.
- Поставь её откуда взял
- Спасибо большое! Продолжай свои дела.
- Кхм… Процессор, видишь перец?
- И так далее....

В какой-то момент это всё задолбало программистов. И они решили: Хей, а почему бы нам просто не не написать инструкцию «Передай мне соль»? Так и сделали. Набор таких комплексных инструкций назвали CISC.

Этот подход стал настоящим спасением как для разработчиков, так и для бизнеса. Захотел клиент новую инструкцию — не проблема, были бы деньги — мы сделаем. А деньги у клиентов были.

Недостатки CISC

Но был ли такой подход оптимальным??? С точки зрения разработчиков — да. Но вот микроархитектура страдала.

Представьте, вы купили квартиру и теперь вам нужно обставить её мебелью. Площади мало, каждый квадратный метр на счету. И вот представьте, если бы CISC-процессор обставил мебелью вам гостиную, он бы с одной стороны позаботился о комфорте каждого потенциального гостя и выделил бы для него своё персональное место.

С другой стороны, он бы не щадил бюджет. Диван для одного человека, пуф для другого, кушетка для третьего, трон из Игры Престолов для вашей Дейенерис. В этом случае площадь комнаты бы очень быстро закончилась. Чтобы разместить всех вам бы пришлось увеличивать бюджет и расширять зал. Это не рационально. Но самое главное, CISC-архитектура существует очень давно и те инструкции, которые были написаны в 60-х годах сейчас уже вообще не актуальны. Поэтому часть мебели, а точнее исполнительных блоков, просто не будут использоваться. Но многие из них там остаются. Поэтому появился RISC…

Преимущества RISC

С одной стороны писать на Assembler'е под RISC процессоры не очень-то удобно. Если в лоб сравнивать код, написанный под CISC и RISC процессоры, очевидно преимущество первого.

Так выглядит код одной и той же операции для x86 и ARM.

x86

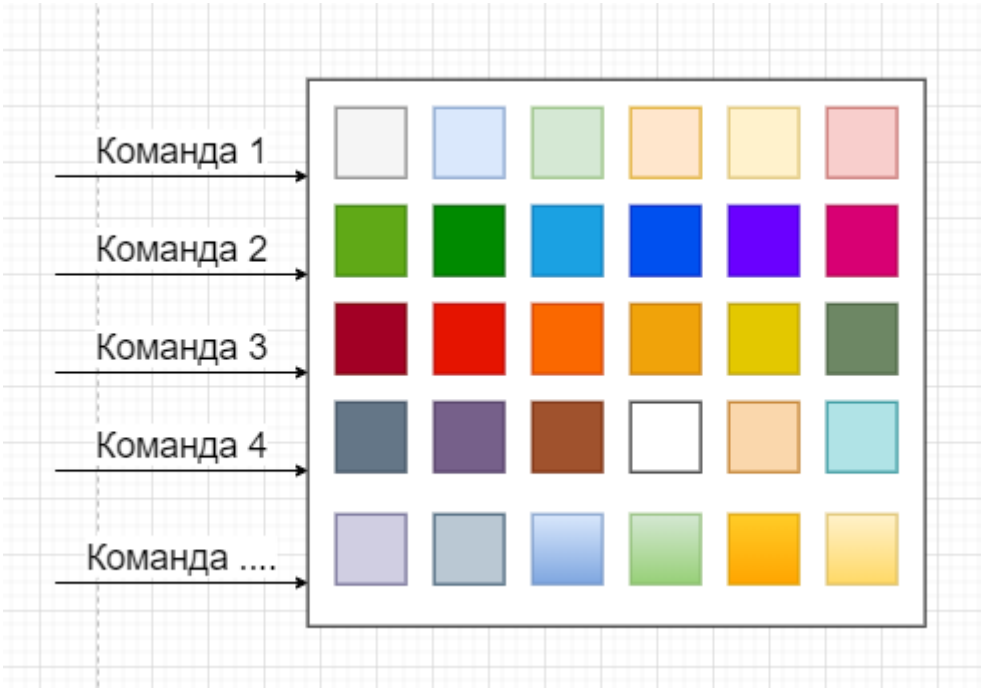
- MOV AX, 15; AH = 00, AL = 0Fh
- AAA; AH = 01, AL = 05
- RET

ARM

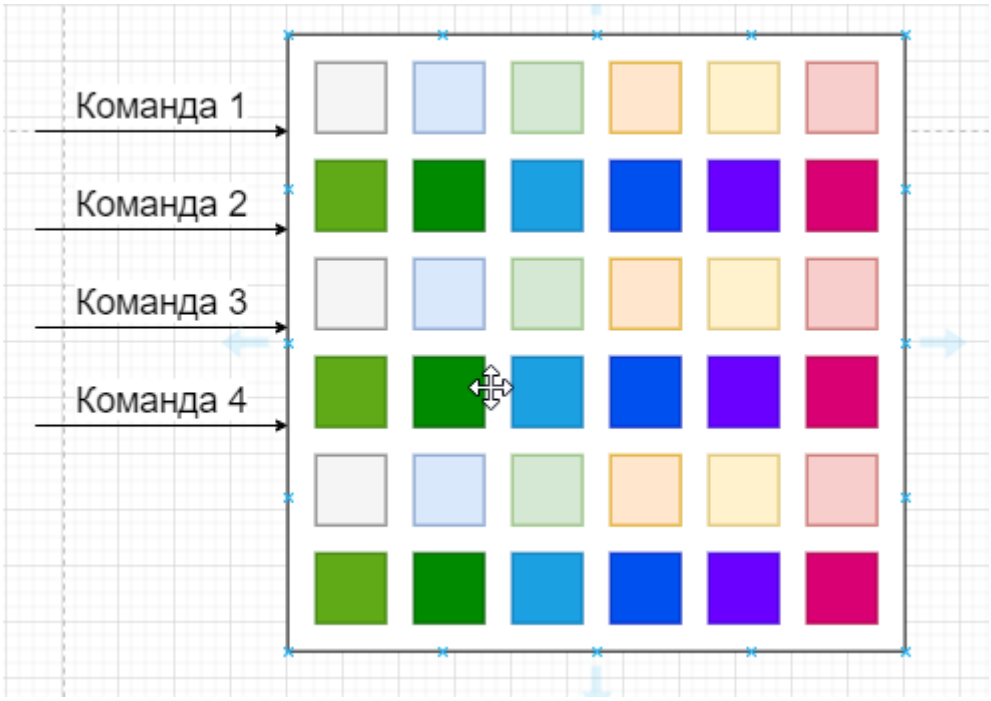
- MOV R3, #10
- AND R2, R0, #0xF
- CMP R2, R3
- IT LT
- BLT elsebranch
- ADD R2. #6
- ADD R1. #1
- elsebranch:
- END

Но так было раньше. На ассемблере уже давно никто не пишет. Сейчас за программистов всё это делают компиляторы, поэтому никаких сложностей с написанием кода под RISC-процессоры нет. Зато есть преимущества.

Представьте, что вы проектируете процессор. Расположение блоков на x86 выглядело бы так.



Каждый цветной квадрат — это отдельные команды. Их много и они разные. Как вы поняли, здесь мы уже говорим про микроархитектуру, которая вытекает из набора команд. А вот ARM-процессор скорее выглядит так.



Ему не нужны блоки, созданные для функций, написанных 50 лет назад.

По сути, тут блоки только для самых востребованных команд. Зато таких блоков много. А это значит, что можно одновременно выполнять больше базовых команд. А раритетные не занимают место.

Еще один бонус сокращенного набора RISC: меньше места на чипе занимает блок по декодированию команд. Да, для этого тоже нужно место. Архитектура RISC проще и

удобнее, загибайте пальцы:

- проще работа с памятью,
- более богатая регистровая архитектура,
- легче делать 32/64/128 разряды,
- легче оптимизировать,
- меньше энергопотребление,
- проще масштабировать и делать отладку.

Для примера вот два процессора одного поколения. ARM1 и Intel 386. При схожей производительности ARM вдвое меньше по площади. А транзисторов на нем в 10 раз меньше: 25 тысяч против 275 тысяч. Энергопотребление тоже отличается на порядок: 0.1 Ватт против 2 Ватт у Intel. Шок.

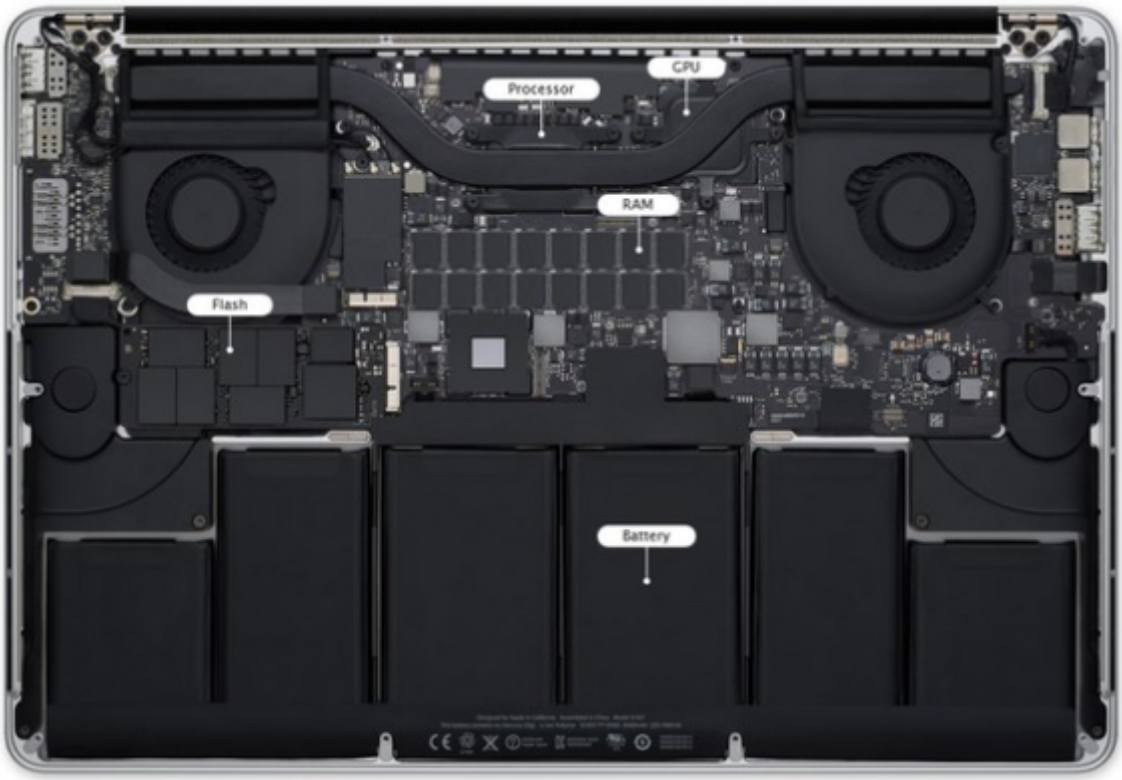
Поэтому наши смартфоны, которые работают на ARM процессорах с архитектурой RISC, долго живут, не требуют активного охлаждения и такие быстрые.

Лицензирование

Но это все отличия технические. Есть отличия и организационные. Вы не задумывались почему для смартфонов так много производителей процессоров, а в мире ПК на x86 только AMD и Intel? Все просто — ARM это компания которая занимается лицензированием, а не производством.

Даже Apple приложила руку к развитию ARM. Вместе с Acorn Computers и VLSI Technology. Apple присоединился к альянсу из-за их грядущего устройства — Newton. Устройства, главной функцией которого было распознавание текста.

Даже вы можете начать производить свои процессоры, купив лицензию. А вот производить процессоры на x86 не может никто кроме синей и красной компании. А это значит что? Правильно, меньше конкуренции, медленнее развитие. Как же так произошло?

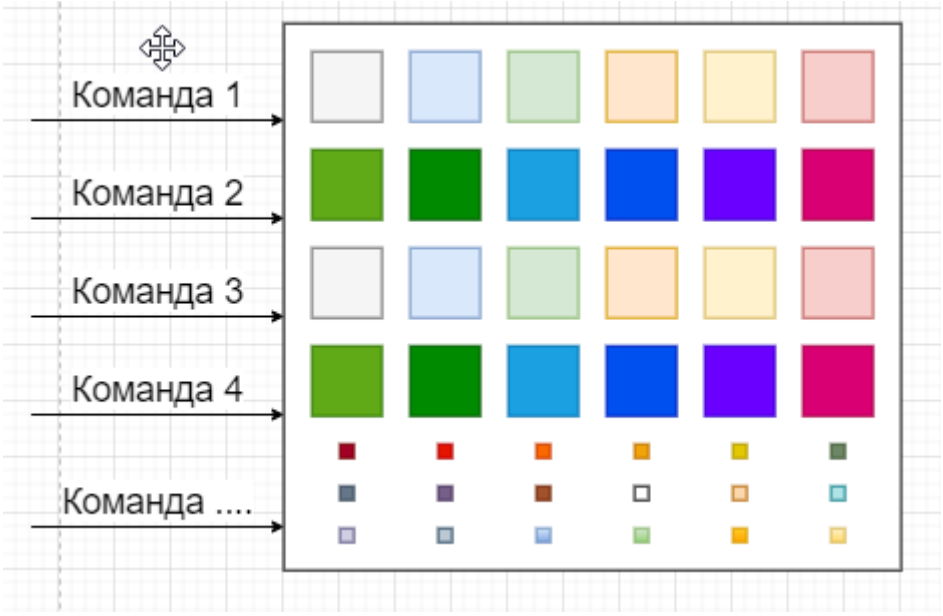


Ну окей. Допустим ARM прекрасно справляется со смартфонами и планшетами, но как насчет компьютеров и серверов, где вся поляна исторически поделена? И зачем Apple вообще ломанулась туда со своим Apple Silicon.

Что сейчас?

Допустим мы решили, что архитектура ARM более эффективная и универсальная. Что теперь? x86 похоронен?

На самом деле, в Intel и AMD не дураки сидят. И сейчас под капотом современные CISC-процессоры очень похожи на RISC. Постепенно разработчики CISC-процессоров все-таки пришли к этому и начали делать гибридные процессоры, но старый хвост так просто нельзя сбросить.



Но уже достаточно давно процессоры Intel и AMD разбивают входные инструкции на более мелкие микро инструкции (micro-ops), которые в дальнейшем — сейчас вы удивитесь — исполняются RISC ядром.

Да-да, ребята! Те самые 4-8 ядер в вашем ПК — это тоже RISC-ядра!

Надеюсь, тут вы окончательно запутались. Но суть в том, что разница между RISC и CISC-дизайнами уже сейчас минимальна.

А что остается важным — так это микроархитектура. То есть то, насколько эффективно все организовано на самом камне.

Ну вы уже наверное знаете, что Современные iPad практически не уступают 15-дюймовым MacBook Pro с процессорами Core i7 и Core i9.

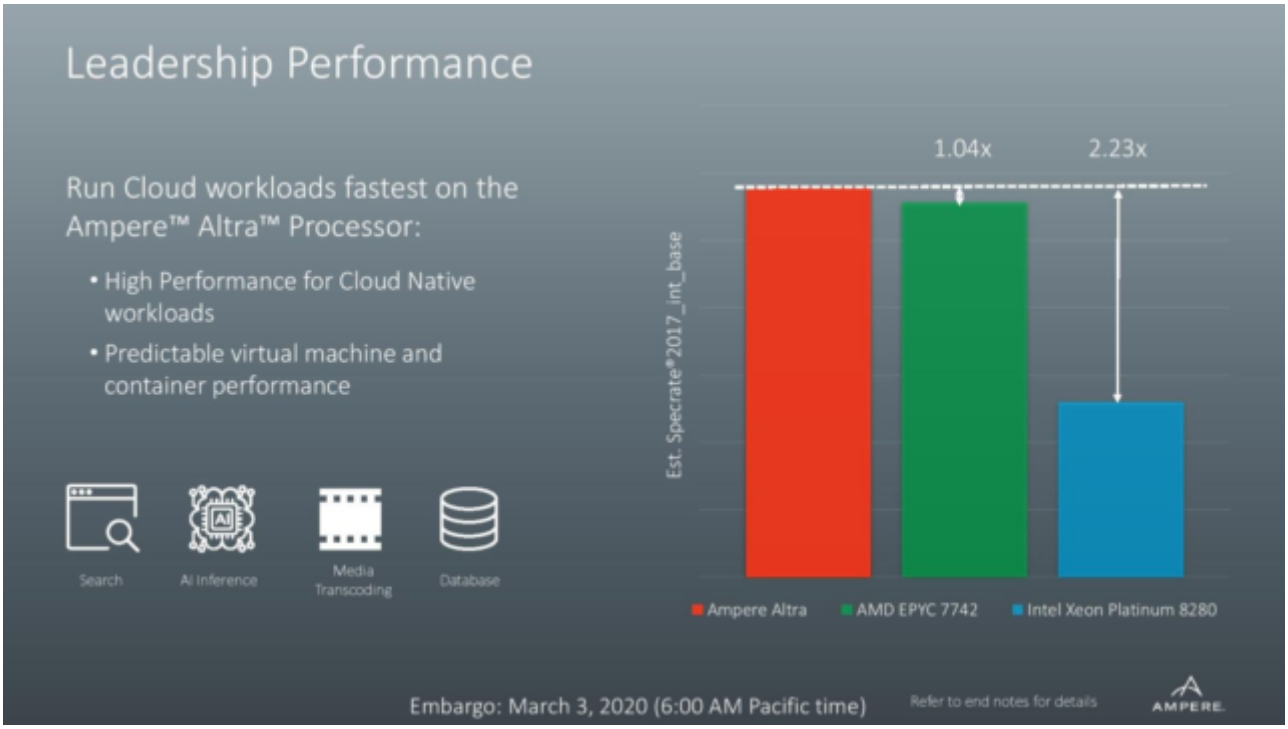
Geekbench Results

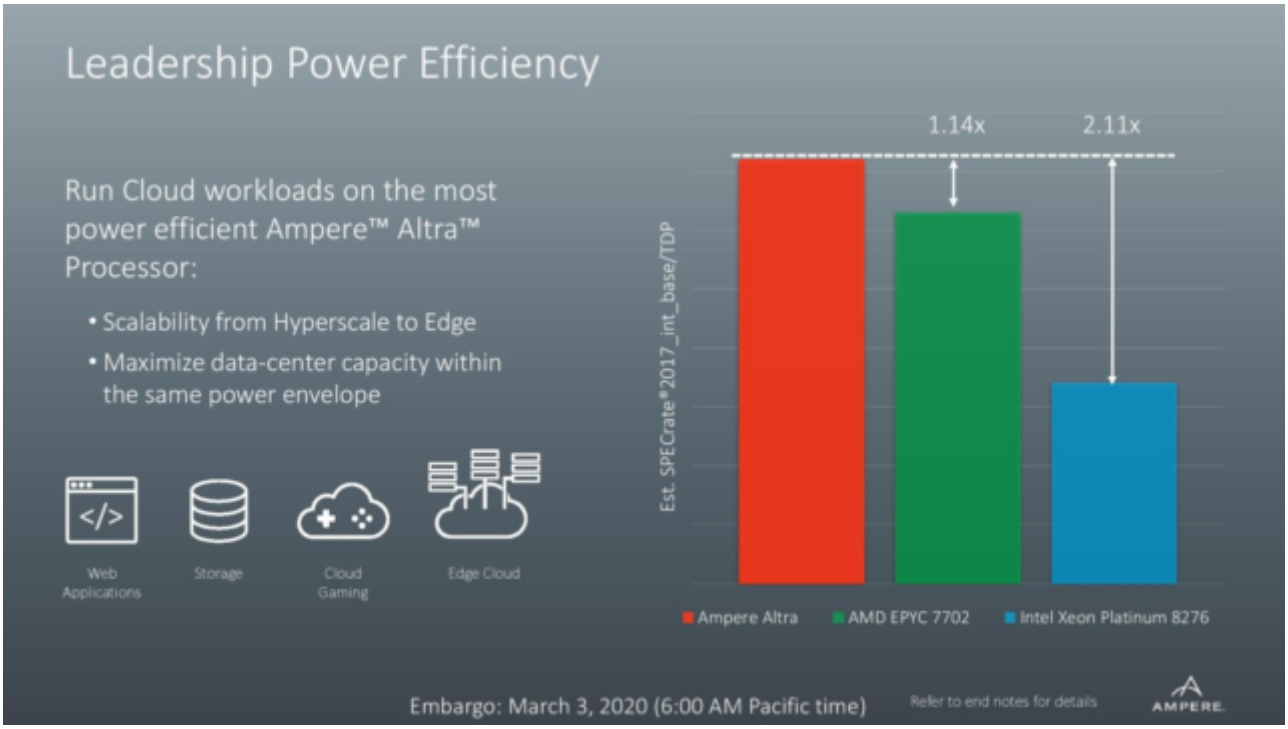
	Chip	Single-Core Score	Multi-Core Score
2018 iPad Pro	A12X	5,025	18,106
iPhone XS / XS Max / XR	A12	4,795	11,167
2017 iPad Pro	A10X	3,913	9,327
2018 iPad	A10	3,474	5,914
2018 15" MacBook Pro	2.2GHz six-core Core i7	4,928	21,165
	2.6GHz six-core Core i7	5,053	21,351
	2.9GHz six-core Core i9	5,344	22,552

MacRumors

А что с компьютерами?

Недавно компания Ampere представила свой 80-ядерный ARM процессор. По заявлению производителя в тестах процессор Ampere показывает результат на 4% лучше, чем самый быстрый процессор EPYC от AMD и потребляет на 14% меньше энергии.





Компания Ampere лезет в сегменты Cloud и Workstation, и показывает там отличные цифры. Самый быстрый суперкомпьютер в мире сегодня работает на ARM ISA. С обратной стороны, Intel пытается все таки влезть в сегмент low power и для этого выпускает новый интересный процессор на микроархитектуре lakefield.

Пока у ноутбуков и процессоров от Intel есть одно неоспоримое достоинство - *(охлаждение и)* единство архитектуры. Пока на рынке ARM-процессоров существуют Qualcomm, Samsung, MediaTek, в мире x86 творится монополия и разработчикам сильно легче делать софт и игры под “взрослые” процессоры.

И Apple та компания, которая способна мотивировать достаточное количество разработчиков пилить под свой ARM. Но суть этого перехода скорее не в противостоянии CISC и RISC. Поскольку оба подхода сближаются, акцент смещается на микроархитектуру, которую делает Apple для своих мобильных устройств. И судя по всему микроархитектура у них крута. И они хотели бы ее использовать в своих компьютерах.

И если бы Intel лицензировал x86 за деньги другим людям, то вероятно Apple просто адаптировали свою текущую микроархитектуру под x86. Но так как они не могут этого сделать, они решили просто перейти на ARM. Проблема для нас с микроархитектурой в том, что она коммерческая тайна. И мы про нее ничего не знаем.

Итоги

Спрос на ARM в итоге вырастет. Для индустрии это не просто важный шаг, а архиважный. Линус Торвальдс говорил, что пока рабочие станции не станут работать на ARM — на рынке серверов будут использовать x86.

И вот это случилось — в перспективе это миллионы долларов, вложенных в серверные решения. Что, конечно, хорошо и для потребителей. Нас ждет светлое будущее и Apple, действительно, совершила революцию!

Редактор материала: Антон Евстратенко. Этот материал помогли подготовить наши зрители Никита Куликов и Григорий Чирков. Спасибо ребята!

Теги: arm, x86, intel, amd, qualcomm, tsmc, samsung, apple, exynos, kirin, huawei, CISC, RISC, микропроцессоры, процессоры, микрочипы, чипы, чипсеты, программирование, компиляторы

Хабы: Блог компании Droider.Ru, Программирование, Смартфоны, Ноутбуки, Процессоры

Droider.Ru

Компания

124 Карма 0 Рейтинг

Валерий Истишев @istishev

Пользователь

Комментарии 197

НЛО прилетело и опубликовало эту надпись здесь

thealfest
18 сен 2020 в 19:24

Mips и Power — это тоже Risc.

0 Ответить

creker
18 сен 2020 в 23:38

Что ничего особо не значит. Внутри все эти процессоры кардинально разные. RISC давно уже ничего не значащая аббревиатура, как и противостояние CISC и RISC, когда и те и другие разбивают инструкции на микрооперации.

+6 Ответить

DrPass
19 сен 2020 в 03:13

Вообще, разница между современными RISC и CISC есть, но она совсем не там, где её видит автор статьи (и чему он, к сожалению, посвятил половину этой самой статьи). От всего того противостояния архитектур сейчас выжил всего лишь один признак: RISC-процессоры имеют в своём наборе команд отдельно команды для пересылки данных между памятью и регистрами, и отдельно команды для операций над данными в регистрах. Команды CISC-процессоров же в своих операндах могут произвольно использовать как регистры, так и непосредственно адреса в памяти. Вот, собственно, и всё. Архитектурно же внутри и RISC, и CISC сейчас устроены схожим образом.

+15 Ответить

НЛО прилетело и опубликовало эту надпись здесь

vvzvlad
18 сен 2020 в 19:50

Ну, есть, на самом деле, на низком уровне. Не конская, но есть. Просто для пользователя x86 ускоряют разными хаками, и ему кажется, что в общем-то, разницы особой нет. Но потом из-за этих хаков всякие интересные уязвимости появляются.

-5 Ответить

НЛО прилетело и опубликовало эту надпись здесь

vvzvlad
18 сен 2020 в 20:24

Ну, так превращать одни команды в другие все равно надо.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

apro
18 сен 2020 в 21:59

Но при этом в процессорах Intel есть кэш декодера микроопераций. В последних по-моему около 2000 элементов. Странно кэшировать то что не тормозит. И в "Intel Tremont" предназначенного для "low-power" этого кэша как раз нет. Зачем выкидывать то что не сильно влияет на энергопотребление, ведь по идее чем более похожим будет новый Atom на остальные CPU тем всем легче.

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

netch80
21 сен 2020 в 19:34

кэш декодера микроопераций

Микрооперации декодировать нет смысла, они уже представлены в целевом виде конкретной модели процессора.
Кэш декодера инструкций (команд) системы x86 назывался trace cache и, насколько мне известно, последняя модель, в которой он был, назывался Pentium 4 — и от него отказались в Core.
Если у вас другие данные, укажите источник.

0 Ответить

tyomitch
21 фев 2021 в 10:02

> Later, Intel included μop caches in its Sandy Bridge processors and in successive microarchitectures like Ivy Bridge and Haswell.[31]:121–123[35] AMD implemented a μop cache in their Zen microarchitecture.[36]
>
> Fetching complete pre-decoded instructions eliminates the need to repeatedly decode variable length complex instructions into simpler fixed-length micro-operations, and simplifies the process of predicting, fetching, rotating and aligning fetched instructions. A μop cache effectively offloads the fetch and decode hardware, thus decreasing power consumption and improving the frontend supply of decoded micro-operations. The μop cache also increases performance by more consistently delivering decoded micro-operations to the backend and eliminating various bottlenecks in the CPU's fetch and decode logic.[34][35]
>
> A μop cache has many similarities with a trace cache, although a μop cache is much simpler thus providing better power efficiency; this makes it better suited for implementations on battery-powered devices. The main disadvantage of the trace cache, leading to its power inefficiency, is the hardware complexity required for its heuristic deciding on caching and reusing dynamically created instruction traces.[37]

+1 Ответить

netch80
21 фев 2021 в 10:55

Насколько я понимаю, официальных утверждений как минимум от Intel про этот кэш нет; информацию находят косвенно и не совсем гарантированно по измерениям скорости, по патентам и т.п.? У всех этих упоминаний нет дальнейших ссылок.
Вообще спасибо, пропустил это — как-то последние годы была привычка больше смотреть в оф. доки, чем на всякие wikichip, после того, как в последних не находилось ответов на вопросы. (В оф. доке — тоже, но она как-то легче вспоминалась.)

0 Ответить

tyomitch
21 фев 2021 в 11:01

2.5.2.2 Decoded ICache

0 Ответить

Alex_ME
18 сен 2020 в 21:22

Не думаю, что корректно говорить, что внутри RISC, потому что внутри мощных RISC'ов (ARM) все равно то же самое происходит. Декодирование инструкций в микрооперации, кэши микроопераций, реордеринг и внеочередное исполнение, векторизация, конвейеризация. И все это твикают. Улучшили чуть-чуть предсказатель ветвлений,

улучшили конвейрезицию запросов к памяти, улучшили кэширование, там, сям, и выжимают те самые проценты прироста производительности. Чтобы ARM имел ту же "производительность" на такт (не говорю инструкцию на так, потому что, к примеру, можно улучшать подсистему памяти ипр), ему потребуются те же хаки и улучшения.

+8 Ответить

creker
18 сен 2020 в 20:09

Какие хаки? Внеочередное исполнение, предсказания, спекулятивное исполнение использовать будет любой процессор, которому нужно быть быстрым. АРМы в том числе, как тот же cortex-A9 и старше. Это очевидные оптимизации. x86 ускоряют выносом все большего числа инструкций в железо, добавкой расширений и постоянными твиками микроархитектуры. Вон на райзены посмотреть, как там достигли таких успехов. А уязвимости появляются из-за того, как это все реализовано. АМД при все тех же оптимизациях большинство уязвимостей миновала. Как и армыминовали большинство, но не все. Спектр всех зацепил.

+10 Ответить

mistergrim
19 сен 2020 в 05:55

> Но потом из-за этих хаков всякие интересные уязвимости появляются.

Ага, то-то Meltdown в ARM обнаружили, а в AMD нет.

+5 Ответить

klirichek
20 сен 2020 в 18:34

Если подумать — Spectre и Meltdown — это следствие "протечки абстракций", но никак не архитектуры в целом.
Процессор дошёл до точки ветвления, зависимой от значения в некоей ячейке памяти — и чтобы не ждать, решил просчитать предсказанную ветвь (или сразу обе). И это момент первой протечки: права доступа не проверяются.
Далее, оказалось, что прав нет. Значит, надо результаты ЛЮБОЙ спекулятивной операции устранять (чистить кэш). Туда ходить было нельзя, значит и кэшировать результаты выполнения любых действий над запрещённой областью тоже нельзя!
Но этого не делается, и вот она — уязвимость. Померил время доступа к паре доступных ячеек, одна из которых уже прочиталась в кэш из-за спекуляции над недоступной памятью — и знаешь, что за битик там стоял…
И всё это — следствие не разницы между x86/arm, а скорее разницы в микрооптимизациях и архитектуры (будь там не намертво втравленная в кремний схема, а ПЛМ, у которой можно поменять алгоритм этого микроулучшения с помощью микропрошивки — и проблемы бы не было).

+3 Ответить

dragonnur
29 сен 2020 в 05:41

Что-то схожее с ПЛМ там есть. В более новых версиях биоса встречались блобы, распаковывающие патчи микроопераций.

0 Ответить

tyomitch
21 фев 2021 в 10:11

> Значит, надо результаты ЛЮБОЙ спекулятивной операции устранять (чистить кэш). Туда ходить было нельзя, значит и кэшировать результаты выполнения любых действий над запрещённой областью тоже нельзя!
> Но этого не делается, и вот она — уязвимость.

Мне очень интересно, как бы вы это реализовали с учётом того, что **каждая** выполняемая операция выполняется спекулятивно. Писать в кэши и в TLB только при retirement? Т.е. пять идущих подряд обращений к одной кэш-строке — будут все пять читаться мимо кэша?

0 Ответить

DistortNeo
18 сен 2020 в 21:30

Я тоже не понимаю этого срача. Есть архитектура, а есть реализация архитектуры в конкретных процессорах. Оценивать архитектуру по процессорам, которые ещё и занимают разные ниши — бесполезная затея.

Современные x86 процессоры прожорливые не потому, что архитектура такая, а потому что они занимают нишу, где производительность важнее энергоэффективности. Уберите из современных x86 процессоров самые прожорливые части — внеочередное выполнение команд и предсказатели переходов, уменьшите размер кэша, понизьте частоту и вольтаж. И получите на выходе процессор, по энергоэффективности сравнимый с ARM-процессорами. Вот только из-за своей производительности он никому не будет нужен.

+5 Ответить

edo1h
18 сен 2020 в 22:43

ну intel пыталась выйти с атомами на рынок телефонов, что-то отношение производительность/энергопотребление не особо оказалось.

+1 Ответить

Grox
18 сен 2020 в 23:38

У меня был Asus Zenfone 2 на Атом 4 х 2ГГц*, мощный и работа от батареи не отличалась от других чем-то особенно.
*Без подключения к зарядке работал по умолчанию на 1,8.

+6 Ответить

eldog
30 сен 2020 в 16:22

Кстати, хороший телефн, я сам пользовался, потом сын таскал, потом дочь, пока экран не разбила.

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

edo1h
19 сен 2020 в 16:25

> …из-за своей немассовости.

вас послушать — ни одна новая технология никогда не займёт свою долю на рынке, она же дороже из-за своей немассовости.

> …с нативным кодом до сих пор компилируются только под ARM и MIPS…

то есть под MIPS разработчикам не лень компилировать, а под x86, на котором они и разрабатывают в 99% случаев — лень?

+2 Ответить

НЛО прилетело и опубликовало эту надпись здесь

edo1h
20 сен 2020 в 05:47

было бы просто не лучше — заняло бы какую-то долю рынка. потому что у всех на слуху (и x86, и intel), потому что разработчикам чуть удобнее (можно отлаживать код прямо на рабочей станции безо всяких эмуляторов), потому что пользователям чуть удобнее (можно запускать статически собранный софт для linux, а в chroot — и не статически)

> Во-вторых, программы для Windows все равно запускать нельзя.

btw, можно (как минимум теоретически), wine и kvm в помощь. и если первое на arm просто не сможет запустить x86 бинарники, то второе сможет, но будет тормозить.

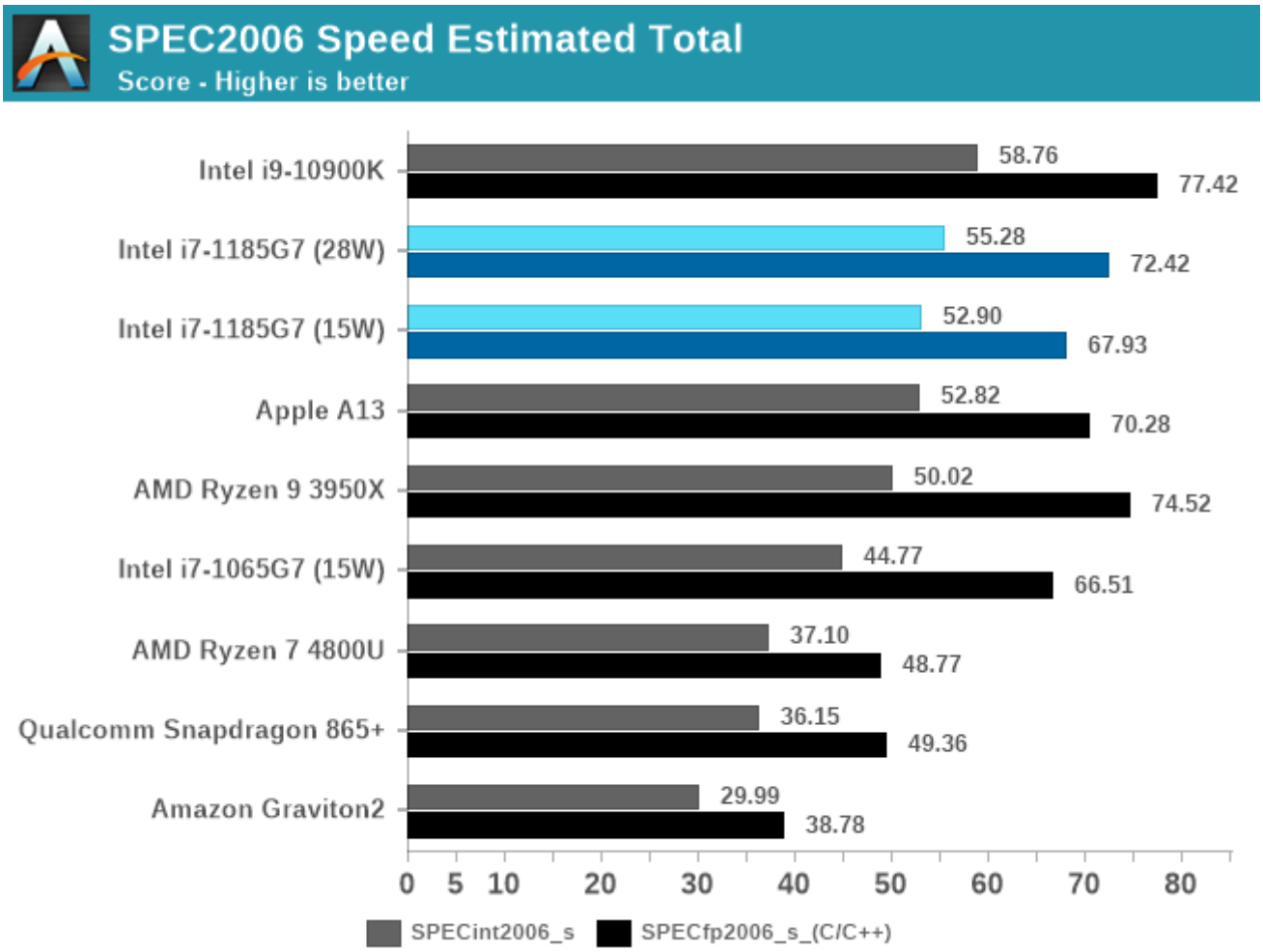
но спроса на x86 нет. настолько нет, что вы сами пишете, что разработчики скорее соберут бинарники под экзотический mips (не помню ни одного устройства на mips, выпущенного в последние годы, за исключением роутеров, разумеется).
так что я делаю вывод, что atom оказался не просто «не лучше», а хуже.

0 Ответить

beeruser
18 сен 2020 в 23:47

> Хватит тиражировать уже эту фигню

Давайте обойдёмся без истерик, а лучше обратимся к объективным данным
anandtech.com/show/16084/intel-tiger-lake-review-deep-dive-core-11th-gen/8



Видно, что однопоточная производительность примерно одинаковая.
А вот если посмотреть на энергопотребление в этом тесте, то картина более интересная:
10900k — 40W (однопоток+турбо 5,3ГГц)
1185g7 — 20W (4,8ГГц)
A13 — 4W (2.66ГГц)
865 — 2W (3.1ГГц)

(цифры от автора)

> выигрыш от десятикратного увеличения быстродействия за те же Ватты

Это работает в обратную сторону: 40W/4W = 10x
Кстати, по имеющейся информации A14 обещает быть на ~16% быстрее -> Score = ~60+

Энергоэффективность идёт от правильной **микро**архитектуры.
Влияние архитектуры не такое значительное (хотя и есть), но производительность явно не зависит от наличия команд типа AAA (кто придумал этот бессмысленный пример?).
Специально для автора:
AAA and AAS are not available in 64-bit code on x86-64 processors
А в Pentium4 эта команда занимала 27 тактов, к примеру.

> Тем более, что при такой разнице быстродействия программная эмуляция была бы сопоставима по скорости с «родными» процессорами.

Так уже :)
browser.geekbench.com/v5/cpu/3804101
browser.geekbench.com/v5/cpu/3027088
В GB5 A12Z набирает ~1100+ попугаев и ~840+ в режиме эмуляции x86.
Т.е. уровень десктопного Haswell/Ryzen 7 1700
A14 на 40% быстрее чем A12.

+8 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Viknet
19 сен 2020 в 15:36

> Где ноутбуки, работающие по многу суток при сопоставимой с i7 производительности?

Процессор — далеко не единственное, что потребляет энергию в ноутбуке. Улучшение энергоэффективности может дать рост пиковой производительности в том же теплопакете, или снизить теплопакет при той же производительности, но кардинального повышения времени автономной работы ожидать не стоит.

> Где сервера с десятками процессорных ядер, каждое из которых потягается с целым современным Xeon-ом?

Не надо передёргивать, никто такого нигде не обещает.

> Почему такая красивая диаграмма остается диаграммой, а предлагаемые редкими хостинг-провайдерами ARM-сервера — гиковские недоразумения с

> производительностью из 2010-х?

Хостинги начинают появляться со вполне нормальнойпроизводительностью, но на продвижение нужно время. Бизнесы не любят бросаться в авантюры по смене архитектуры не оценив затрат и рисков.

Нормальные ноутбуки на ARM начнут появляться в конце этого — начале следующего года. Чтобы выпустить удачный потребительский ноутбук, голой производительности процессора маловато, нужно как минимум решить вопросы доступности софта.
Именно это мешает какому-нибудь Surface Pro X стать популярным устройством.

0 Ответить

NetBUG
19 сен 2020 в 15:54

Ну была Toshiba A100 на тегре. По ощущениям, сливала eeePC на атомах.

+6 Ответить

Viknet
19 сен 2020 в 16:01

Да, в то время ARM не мог тягаться по производительности с x86, а x86 с трудом влезал в теплопакет мобильника (хотя были отдельные девайсы вроде Nokia 9000/9110).
С тех пор ситуация немного изменилась.

0 Ответить

NetBUG
19 сен 2020 в 18:39

Были)
Справедливости ради, тошиба на тегре – это скорее 2009 год, а не 1998.
Потом у меня был Asus TF700, на котором убунту адово тормозила, но это вопрос скорее про оптимизацию ОС

0 Ответить

Viknet
19 сен 2020 в 20:44

У меня был Asus T91mt, на вполне себе атоме, но пользоваться им было крайне мучительно как под Windows, так и под собранным с оптимизациями Gentoo.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

khajiit
19 сен 2020 в 19:56

Голая производительность уже у них на уровне. Дайте нормальную шину к памяти, а не сравнивайте 128bit@4800 DDR4 с 16bit@1333 DDR3, чтобы успевать прокачивать данные, да дата-кешей побольше — и случится закономерное чудо.

-1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

khajiit
19 сен 2020 в 20:45

3 ГГц — вполне нормальная частота, 5ГГц любой ценой оставим синим.
Ну и кроме голой частоты на производительность влияет еще множество факторов.
И, в целом, потолок у армов видится более высоким при том же потреблении.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

khajiit
19 сен 2020 в 22:58

Скорее, не для серверов, а для x86. Современный CPU — это связка из бутылочных горлышек, и каждое лишнее (транслятор, да) ухудшает быстродействие в целом и порождает артефакты при взаимодействии с другими. Плюс у армов есть Thumb, который увеличивает плотность кода, а с ним и максимально достижимую скорость обработки команд и населенность кешей инструкциями, условное выполнение, позволяющее вообще не сбрасывать конвейер независимо от удачи или неудачи предсказания, что обратно упрощает конвейеры…

Ну и наличие вменяемой конкуренции здорово сказывается на потребителе) А линуксам все равно, на какой архитектуре работать.

+1 Ответить

BD9
19 сен 2020 в 23:04

Под ARM Linux работает, но совсем неоптимально.

-1 Ответить

creker
20 сен 2020 в 00:00

> Плюс у армов есть Thumb

У 64-битных нету.

+1 Ответить

khajiit
20 сен 2020 в 14:30

Да, и вправду, можно использовать только в режиме aarch32, что не поддерживается тем же Apple…
Видимо с переходом к 64-бит адресации решили не экономить, по крайней мере, пока.

0 Ответить

Viknet
19 сен 2020 в 20:04

> сервера с десятками процессорных ядер, каждое из которых потягается с целым современным Xeon-ом"
>
> Уже два стартапа пообещали за этот год.

Поделитесь именами этих стартапов, желательно со ссылками на их заявления.

> Нет. Ноутбуки от Эпл начнут появляться.

А ноутбуки Apple уже не считаются что ли?

> Будут ли они нормальными или уровня слабых процессоров — можно лишь гадать.

Не нужно гадать, чтобы понять, что в ноубуках не будет процессоров слабее, чем в телефонах. А текущий телефонный процессор уже затыкает за пояс всю U линейку Intel.

> А так то ноутбуки на арме уже вроде как были. Что в них было не нормального?

Вы же сами строчкой выше и ответили — слабые были процессоры.
Изменилось это буквально в последние 2-3 года.

-2 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Viknet
19 сен 2020 в 20:39

> Извините, но поиск на хабре работает.

На такой булшит никакой поиск не работает. Никто не делал заявлений, что одно ядро какого-то ARM тягается с многоядерными Xeon.

> В продаже уже есть, что рвёт i7 или R7 как тузик грелку или нет?
> Если нет, то значит пока ещё ничего нет.

Вы с чем спорите, с голосами в голове? Я писал, дословно, "Нормальные ноутбуки на ARM начнут появляться в конце этого — начале следующего года."

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Viknet
19 сен 2020 в 21:01

> И пока по тестам того, что есть — это где-то i3.

Пока по тестам того, что есть — это где-то на одном уровне с топовыми U-процессорами как в однопотоке, так и в многопотоке:
https://browser.geekbench.com/processors/intel-core-i7-8569u
https://browser.geekbench.com/ios_devices/ipad-pro-12-9-inch-4th-generation
Только с разницей в тепловыделении в 4 раза.

> человек утрировал

Поэтому я и написал, что не надо передёргивать. После чего вы влезли утверждая, что аж два стартапа такое заявляли.

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Viknet
19 сен 2020 в 21:40

> 8 поколение — давно уже не сильно топовое, уж два года прошло.

Ок, не заметил, что Intel замудрила маркировку своих "экономичных" процессоров. Но результат от этого не сильно поменялся.
Если открыть каталог Intel потом взять топовые индексы по каждому постфиксу и выбросить всё, что 45+ Вт (и жалкий 7 Вт i7-10510Y), то остаётся вот такой набор.
Из них только для i7-1068ng7 есть сводная статистика, но можно и поиском воспользоваться (1, 2), чтобы увидеть не очень впечатляющие результаты.

Да и A12Z по факту тот же двухлетний процессор A12X с ещё одним графическим ядром.

> Тут и 10 нм от интела, и 7 нм от амд.
> Причём там ядер больше

Я не нашёл в каталоге Intel 10 нм процессора с больше, чем 4 ядрами.
6 ядер на 14 нм не сильно помогли: i7-10810U

> Поэтому даже если производительность ±, то вот потребление снизилось капитально.

Но всё ещё далеко до реальных 7-8 Вт A12Z.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Viknet
19 сен 2020 в 22:12

> Вот за счёт более дорогого техпроцесса ЭПЛ может что и сделает, только кто сказал, что те же АМД на 5 нм не окажутся сильно лучше?

У них не только техпроцесс, но и своя микроархитектура, которая, похоже, довольно хороша.
AMD сегодня на тех же 7нм не догоняет позапрошлогодный Apple в производительности на ядро при многократном энергопотреблении.

Я вообще с надеждой смотрю в будущее. Скорее всего все будут двигаться примерно в одном направлении, и усиление конкуренции на рынке процессоров для ноутбуков и десктопов ещё одним игроком в виде Apple пойдёт только на пользу конечным потребителям, как с прорывами AMD в последние 2-3 года.

0 Ответить

iproger
20 сен 2020 в 05:25

У Амд очень неплохие ядра, но у них есть проблема в виде чиплетного дизайна. Не совсем честно сравнивать их в таком виде который эффективен в экономическом смысле.

0 Ответить

Viknet
20 сен 2020 в 10:39

Почему не честно? Мы цены тут совсем не рассматриваем, только производительность в каком-то условном теплопакете.

0 Ответить

iproger
20 сен 2020 в 18:54

Потому что компания сознательно идет на удешевление чипов которые потенциально могут больше. Возможно, в будущем сделают все на одном кристалле без лишних задержек и потерь на соединении чипов, тогда увидим что может их архитектура. А пока что можно посмотреть только новые ноутбучные процессоры.

0 Ответить

creker
20 сен 2020 в 19:02

Она на это идет, чтобы быть лучше интела, который как раз с монолитными кристаллами уперся в потолок на серверном рынке. Первый зен и так был монолитным более менее кристаллом, но внутри все так же была infiinity fabric. Последняя самое главное достижение их архитектуры и было бы странно видеть отказ от этого подхода. Кристаллы дешевле не становятся, наоборот чем тоньше процесс, тем дороже выходит. Производительность однопотока сильно не растет, а ядер много в один кристалл не засунешь. Интел то уж знает. Вот и выбрала амд путь, который дал им победу.

0 Ответить

Viknet
20 сен 2020 в 21:34

Так же можно сказать, что Apple сознательно идёт на ухудшение производительности, чтобы телефоны не обжигали руки. А Intel сознательно не идёт печататься к TSMC, чтобы сохранить контроль над производством, потеряв сегодня в техпроцессе.
Но это их внутреннее дело, мы сравниваем не то, что компании могли бы выпустить в теории, а реальные продукты, которые можно сейчас купить.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Antervis
20 сен 2020 в 03:34

> Процессор — далеко не единственное, что потребляет энергию в ноутбуке

практически единственное. Процессор в ноутбуке жрет до 15/25/45/65 ватт в зависимости от класса ноутбука и утилизации, плюс дискретка в игровых, экран — пару ватт, wifi полтора-два, bluetooth еще меньше, а что еще может жрать? Ну, I/O а-ля thunderbolt, но обычно подключенный к монитору ноутбук еще и на зарядке. В итоге даже в ультрабуке процессор будет составлять порядка 2/3 от общего энергопотребления.

0 Ответить

Viknet
20 сен 2020 в 10:43

> Процессор в ноутбуке жрет до 15/25/45/65 ватт

Он такие цифры жрёт только под нагрузкой, при которой автономная работа любых ноутбуков плачевная, батареи больше 100 Вт*ч практически никто не ставит.
В спокойном режиме, в котором как раз все сравнивают автономность что ноутбуков, что телефонов-планшетов, мобильный процессор может потреблять от единиц до долей ватта.

+1 Ответить

DistortNeo
20 сен 2020 в 11:09

Как раз экран — самое прожорливое. Пару ватт он жрёт, если яркость выставлена на минимум, а при нормальной яркости и с диагональю 15.6" — больше 5 Вт так точно.

Процессор тоже жрёт много, но только под нагрузкой. А постоянная нагрузка процессора — не самый частый сценарий использования ноутбука при работе от батареи. В среднем он потребляет пару Вт при обычной офисной работе.

+4 Ответить

Areso
20 сен 2020 в 19:04

> Процессор — далеко не единственное, что потребляет энергию в ноутбуке. Улучшение энергоэффективности может дать рост пиковой производительности в том же теплопакете, или снизить теплопакет при той же производительности, но кардинального повышения времени автономной работы ожидать не стоит.

Не единственное, но, пожалуй, самое прожорливое (под нагрузкой).

Понятно, что есть чипсет, что есть ОЗУ, что энергонезависимая память, но когда у меня начинает активно нагружаться процессор, компьютер начинает греться как печка, а процессорный вентилятор — выть. И оценка автономной работы падает раза в два-три.

+1 Ответить

DistortNeo
20 сен 2020 в 19:18

А я зарезал TDP до 15W, и когда у меня начинает нагружаться процессор, то вентилятор только включается и начинает слегка шуршать. При этом разница в

производительности по сравнению с TDP 35W незначительна. И если экран работает постоянно, то процессор в таком режиме — всего несколько секунд.

Аналогично со смартфоном: высокая яркость экрана заметно быстрее высаживает батарею.

0 Ответить

Areso
20 сен 2020 в 19:55

Ну, видимо зависит от профиля нагрузки. У меня на корпоративном ноуте постоянно либо браузер выжирает ЦПУ, либо корпоративные AV/DLP системы. Либо вместе.

0 Ответить

Viknet
20 сен 2020 в 21:26

Как я упоминал выше, автономную работу под нагрузкой никто не пытается замерять, потому что она у всех устройств сейчас не очень впечатляет.
Даже айфон можно заставить сесть за 2-3 часа, хорошо нагрузив.

0 Ответить

DistortNeo
19 сен 2020 в 16:58

Эти графики как раз показывают, что процессоры достигли технологического предела в однопотоке. Для процессора Intel увеличение производительности на +14% обернулось ростом энергопотребления в 2.5 раза. Ну и сам тест, судя по всему, вообще не задействует векторые операции.

+1 Ответить

saege5b
19 сен 2020 в 21:48

А можно ещё для сравнения их же, но при сжатии видео в н265, режим- ультра, с полностью перелопаченными настройками, включая какие-нибудь таблицы квантизации, 5 проходов? или хотябы ффмпег с фильтрами: де-эхо, удаление тишины, фильтры высоких и низких частот?
ПыСы Asus ROG Phone3 описание охлаждениявызывает уважение.

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

wataru
20 сен 2020 в 12:27

Вот я чего не понимаю: а почему нельзя этот A13 разогнать в полтора раза, поставить нормальную систему охлаждения (пусть он даже будет вместо 1.5*4W жрать все 40W)? Будет процессор, который на 40% быстрее самого быстрого X86, если верить этому графику.

Почему бы не сделать так (хотя бы в тесте) и не хвастаться везде абсолютно самим быстрым процессором и закрыть спор раз и на всегда?

+2 Ответить

Viknet
20 сен 2020 в 13:10

Потому что самому эпплу неинтересны такие пиписькомерки в отрыве от реальных устройств, они не продают эти чипы никому.
Реальные устройства с большим теплопакетом пока готовятся.

0 Ответить

creker
20 сен 2020 в 13:52

А кто сказал, что его можно разогнать? У процессоров есть технологические и микроархитектурные пределы, выше которых не прыгнешь. У всех процессоров они разные.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

wataru
20 сен 2020 в 15:36

Греться будет слишком сильно. Но у ARM с его в 10 раз меньшим потреблением таких проблем быть не должно.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

DistortNeo
20 сен 2020 в 16:04

> Там с разгоном частоты потребление растёт квадратично.

Квадратично относительно напряжения и линейно относительно частоты.
Если для +20% частоты придётся поднять напряжение с 0.8V до 1.5V, то потребление вырастет больше, чем в 4 раза.

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

creker
20 сен 2020 в 18:31

Проблема не в нагреве. Банально нестабильно работают ядра на высоких частотах. В процессорах все намного сложнее, чем просто повысил частоты. Микроархитектура всегда играла важную роль, почему интел всегда имел частоты выше, но при этом за счет IPC амд могла быть быстрее. Так было во времена атлонов, так обстоит дело сейчас. APM я думаю точно так же упрется в пределы архитектуры конкретного процессора. Ну и тепловыделение конечно, куда без него. Вся энергоэффективность улетит в окно.

0 Ответить

khajiit
20 сен 2020 в 22:30

У них наверняка и частотный потолок будет ниже. Он определяется огромным количеством вещей, от варианта техпроцесса (Haswell как пример) до межсоединений, которые могу оказаться просто не приспособлены к работе на высоких частотах.
Не забывайте, что ARM только начали осваивать частоты, до этого они много лет были жестко оптимизированы под минимальное потребление.

Ну и, как заметили ниже, тупой разгон — бесперспективен, к нему прибегают только когда ничего больше сделать нельзя. Вырвать +5% производительности ценой +30% жора.

+1 Ответить

hogstaberg
22 сен 2020 в 04:21

То есть вы утверждаете следующее: в intel и amd работают настолько бездарные инженеры, что их разработки можно вот так вот взять и обставить практически в 10 раз по энергоэффективности?
Звучит как-то сомнительно. Вот прямо максимально сомнительно.

+1 Ответить

Antervis
20 сен 2020 в 04:17

> Хватит тиражировать уже эту фигню про конскую разницу в энергоэффективности. Нет ее, иначе все давно бы сидели на ARMax и даже отсутствие обратной совместимости с x86 не омрачило бы выигрыш от десятикратного увеличения быстродействия за те же Ватты.

разница есть, не десятикратная конечно, но всё еще «конская». Просто ARM производители раньше осваивали преимущественно развивающийся мобильный/embedded рынок, и только сейчас, когда он насытился, начали пытаться заходить в те сегменты, где доминирует x86. Короче ---сейчас мы здесь---, вот это вот всё

> Тем более, что при такой разнице быстродействия программная эмуляция была бы сопоставима по скорости с «родными» процессорами.

эмулировать x86 мешают в первую очередь патенты intel

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Antervis
20 сен 2020 в 05:38

> И если внутри арма такое крутое ядро, то почему интел или амд не вкрутит его в свои камни и не получит [почти] такую же эффективность? Вот сугубо с теоретической точки зрения, а?

хех, потому что интелу надо поддерживать кучу легаси инструкций, инструкции переменной длины и всякие AVX512 (а в ARM SVE только недавно появился), и всё это задизайнено по принципу «исторически так сложилось» и никак не накладывается на нормальную структуру? Ну это так, предположения. Хотя на самом деле достаточно причины «потому что на переправе коней не меняют».

0 Ответить

DistortNeo
20 сен 2020 в 11:11

Декодеры инструкций занимают в процессоре жалкие проценты от площади кристалла.

+2 Ответить

edo1h
20 сен 2020 в 06:08

> И если внутри арма такое крутое ядро, то почему интел или амд не вкрутит его в свои камни и не получит [почти] такую же эффективность? Вот сугубо с теоретической точки зрения, а?

хороший вопрос.

сначала хотел ответить: не нужно, производительность устраивает, а энергопотребление на десктопах и серверах не так важно.
потом вспомнил про многоядерные серверные процессоры, которые снижают частоту под нагрузкой, то есть высокое потребление снижает производительность, таки оно важно (я

уже молчу про проблемы с охлаждением и потреблением датацентров). да и ноутбучный рынок очень заметен, там экономичные процессоры очень важны.

следующая мысль была: ничего из этого не получится, слишком они разные. но так уже в x86 инструкции разбирает/выполняет микрокод, «скрестить» его с исполняющими блоками arm, наверное, сложно, но возможно.

последняя идея: intel за счёт использования идеологии чучхе получает сверхприбыли, проблем до недавнего времени у него не было, зачем ему было тратиться на лицензирование ядер arm? да и сейчас логичнее выйти из анабиоза и сделать своё ядро, благо ресурсы позволяют.

P. S. согласен, что предположение «если у арма такое крутое ядро» может оказаться неверным, но пока всё выглядит так, будто у сегодняшнего arm существенно меньшее потребление при примерно той же производительности на поток.

-1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

ShadowMaster
18 сен 2020 в 17:22

Вроде это хабр, а разжевывают как для школьников младших классов.
x86(-64) давно CISC только снаружи, внутри же RISC, начиная примерно с Pentium Pro.
С лицензированием какие проблемы? Купить AMD или VIA, денег у Apple вагон.

+17 Ответить

aram_pakhchanian
18 сен 2020 в 19:18

Это очень будет довольно тоскичная покупка. Ну и не покупают же компании, потому что денег — вагон.

0 Ответить

ShadowMaster
18 сен 2020 в 19:35

Это было вот к этому.

> И если бы Intel лицензировал x86 за деньги другим людям, то вероятно Apple просто адаптировали свою текущую микроархитектуру под x86. Но так как они не могут этого сделать, они решили просто перейти на ARM.

Если реально Apple требуется лицензия на x86 архитектуру, то можно было бы прикупить одного из трех обладателей. Видимо решили, что своя разработка принесет больше прибыли. Скоро увидим.

0 Ответить

thealfest
19 сен 2020 в 08:55

Элементы RISC архитектуры были уже в 486.

0 Ответить

NIKOSV
20 сен 2020 в 05:16

AMD почти $100 лярдов стоит, поздно покупать даже для эпла, надо было 5 лет назад покупать когда AMD было на грани банкротства.

0 Ответить

ShadowMaster
20 сен 2020 в 17:41

Первые слухи о скором переходе на ARM были еще в 2016 году. Так что время было, желания не было.

0 Ответить

FForth
18 сен 2020 в 17:25

А, есть вероятность, что на уровне микроархитектуры применяется MISC ядро?
Как пример многоядерного-асинхронного MISC контроллера — GA144 (асинронное переключение ядер ~700МГц у данного кристалла с минимальным потреблением в работе)

0 Ответить

edo1h
18 сен 2020 в 17:59

> x86 процессоры используют сложный набор инструкций, который называется CISC — Complex Instruction Set Computing.
>
> ARM процессоры наоборот используют упрощенный набор инструкций — RISC — Reduced Instruction Set Computing.

мгимо финишд, блин. и помимо этого в статье полно ляпов. мусор.

+7 Ответить

krizhanovsky
18 сен 2020 в 18:07

> Но так было раньше. На ассемблере уже давно никто не пишет. Сейчас за программистов всё это делают компиляторы, поэтому никаких сложностей с написанием кода под RISC-процессоры нет.

Вообще-то, пишут. И иногда на CISC асемблере писать проще, чем на C. Посмотрите ядро Linux или более или менее быструю криптобиблитеку (OpenSSL, WolfSSL, Tempesta TLS). Простой пример: как сложить два 128-битных числа (каждое из которых по 2 long'a)? На C вам прийдется делать телодвижения с битом переноса, а на ассеблере у вас уже есть инструкция, которая в старшее 64-битное число добавит перенос со сложения прошлых 64-х битных чисел. Но, с другой стороны, те, кто упирается в бенчмарки, погемороятся с RISC ассемблером.

Я не очень хорошо знаком с ARM, но интересует что у этой архитектуры с virtual APIC? На x86 без vAPIC сеть в виртуалке очень тормозит.

Слышал от знакомых жалобу на другую проблему с APIC на каких-то ARM процессорах: на 8-и ядерном ARM (не знаю точно каком) нельзя было разбросать очереди прерываний с сетевой карты на разные ядра — все прерывания уходили на одно ядро. Как у современных ARM (которые по 128 потоков и выше умеют) дела с APIC?

+4 Ответить

yleo
19 сен 2020 в 02:20

Саня, у Штеуда с APIC-ами исторически лучше было. Они это выстрадали, начиная с конца 90х и где-то до середины нулевых. У ARMов в среднем по больнице с этим плохо, но (буквально, скорее всего) если взять не "телефонный" ARM, а "серверный" (т.е. рассчитанный на виртуализацию), то всё будет принципиально лучше.

Тем не менее, если есть зажать на прерывания, то всё равно тормозит (даже с "тремя" VAPIC) в сравнении с DPKD/Netmap/Seastar ;)

+1 Ответить

krizhanovsky
19 сен 2020 в 22:30

Привет!

В среднем по больниц, понятно. Я мимоходом смотрел, например, на Marvell ThunderX2, которые до 256 потоков имеют, и как-то не понятно как там с APIC, а что с виртуализацией — тем более. Работал с ними или смотрел подробнее? У Xeon с vAPIC не очень — 4 машины (entry level) взяли и ни у одной не было vAPIC полного, чтобы KVM не воткнула на I/O.

С сетью у нас интересная история. С packet forwarding/filtering на уровнях L2-L4 все понятно — логики мало и она относительно легко скейлится на ядра, DPDK и пр. kernel bypass рулят. У нас же HTTPS прокси и в perf top, как правило, HTTP parser, криптография, различная прикладня логика, например генерация cookie или javascript челленджей, матчинг HTTP

заголовков опять же. Т.е. сетевых функций Linux networking мы не видим. Но при, этом без vAPIC в KVM и мы и Nginx "втыкаем" по полной и разница по производительности 2 раза в лучшем случае.

Кстати, я недавно писал про user-space TCP в приложении для HTTP. У Seastar реализация TCP фейковая (в смысле "fake it before you make it") — просто, чтобы поставить рекодные бенчмарки: очень мало кода и с кучей TODO. F-stack показывает хорошие результаты, но я проверил сделали ли они что-то с известной проблемой хеш таблицей TCP соединений (она же давно была описана CloudFlare), но и в ядре Linux, и FreeBSD, и F-stack реализация все та же и примерно с одним подходом.

Итого, несмотря на отличные бенчмарки и безумное число ядер ARM, если у них нет хорошего APIC — с сетью, и думаю, с дисковым I/O тоже, все будет плохо. Нет поддержики виртуализации (в т.ч. virtual page talbe) — все плохо будет в облаках. И что останется? Числодробилки для ML?.. Возможно, и это ложится в карту NVIDIA.

0 Ответить

yleo
20 сен 2020 в 00:47

Видимо я тебя не понял, ибо мне кажется что вы не просто бьётесь с мельницами, а сначала их шевелите и только потом бьётесь ;)

--

Приличный NIC позволит через MSI генерировать любые 2 прерывания для каждого dma-ring в зависимости от его наполнения (не пуст/заполнен больше половины и т. п.), для local APIC любого приличного процессора (с поддержкой PCI-E). Далее, приличный CPU (либо его серверный мост либо MSI-контроллер) позволит из обработчика прерывания прочитать битовую маску актуальных «взведенных» IRQ. Это примерно оптимальный режим обработки прерываний связкой «приличных» NIC+CPU.

При наличии гипервизора требуется чтобы он понимал «приличное железо» и умел с ним взаимодействовать (настраивать NIC и контроллер MSI) чтобы исключить какую-либо возню с IRQ-масками на пути от железа к гостевой системе. Соответственно https://www.linaro.org/blog/kvm-pciemsi-passthrough-armarm64/ и т.п.

--

Но в нагруженной системе старый-добрый NAPI (с приличными драйверами приличного NIC) должен быть не хуже (если не лучше) всей вышеописанной магии, главное чтобы dma-ring действительно пробрасывался (а не перекладывался) в гостевую систему. И вот тут мне на ум приходят упомянутые ветряные мельницы ;)

0 Ответить

krizhanovsky
20 сен 2020 в 18:23

Про ветярнные мельницы я не понял. Мы бьемся за быстрый HTTPS. Для нас "нагруженной системой" может быть, как массивный bare metal, так и однопроцессорная VM, которая тоже должна работать быстро. Для нас, как и большинства людей, не очень приемлемо молотить все доступные CPU на 100% даже при простое.

С теорией NAPI и сетевых прерываний я знаком, но за ссылку спасибо :) Вопросов все равно много остается: проборс VF с VM и SR-IOV — это один только из кейсов, а что будет с трафиком между двумя VM на одном хосте? Будет куча VM-EXIT и в `perf kvm stat` будут `EXTERNAL_INTERRUPT` .

Поживем, увидим как будут развиваться серверные ARM, но пока у них use cases очень ограничены.

+1 Ответить

yleo
23 сен 2020 в 22:21

> Вопросов все равно много остается: проборс VF с VM и SR-IOV — это один только из кейсов, а что будет с трафиком между двумя VM на одном хосте? Будет куча VM-EXIT и в perf kvm stat будут EXTERNAL_INTERRUPT.

Трафик между двумя VM неплохо разруливал-бы 1Hippeus, но в 2014 его посчитали ненужным (включая массу простых и разумных предложений по всяким mailboxes для VM<->HV и VM<->VM, и т.п.).

Тем не менее, если без "бы", то для коммуникаций VM->VM все равно потребуется дешевый и безопасный (не ломающий гипервизор) wake из VM. Пока у меня нет сведений о каких-либо подвижках в железе для акселерации этих моментов (Штеуд погряз в spectre и тех-процессе, другие сюда пока не смотрят).

Соответственно, без такого wake придется дергать гипервизор, в лучшем случаи с учетом порогов/гистерезисов. Но пока (вроде-бы, не знаю) никто не считает это проблемой заслуживающей переделки кодовой базы (virtio и т.д.). Поэтому vSwitch (сам-по-себе он не плох) и разумные страданья за виртуализацию.

Т.е. мне более-менее понятно что и как следует делать (собственно могу), но не понятно кто и как все это организует/возглавит/оплатит. По ощущениям (имею право быть не правым) Штеуд скатывается в некукопожатные по темам касающимся API/BAPI и взаимодействия (ибо "решето"). MIPS — ищут куда продать куки (им не до концептов), ARM (вот на покупателя), RISCV (сильно разрозненно). Может быть у наших в МЦСТ дойдут руки, если их не отшибут "эффективные менеджеры" и давление диванных хейтеров.

0 Ответить

kovserg
18 сен 2020 в 18:47

Главное отличие в длине команды: в ARM она фиксирована, а в x86 она разная.

+2 Ответить

thealfest
19 сен 2020 в 08:57

и это сильно усложняет архитектуру x86\x64.

+2 Ответить

creker
19 сен 2020 в 15:53

Я думаю это усложняет разве что фронтэнд, который у современных процессоров ничтожную площадь кристалла занимает.

0 Ответить

SergeyMax
19 сен 2020 в 11:29

> Главное отличие в длине команды: в ARM она фиксирована, а в x86 она разная.

В результате операция add eax, 12345678h, не помещающаяся ни в два, ни в четыре байта опкода, на ARM превращается в целый квест)

+4 Ответить

cypok
18 сен 2020 в 19:14

> Так выглядит код одной и той же операции для x86 и ARM.
>
> ▸ Заголовок спойлера

Что вообще хотел сказать автор в этом параграфе?..

+5 Ответить

alexxz
18 сен 2020 в 19:35

Видимо, что в ARM нет операций для Binary-Coded Decimal значений. И надо писать код. Имхо, жуткий формат, когда число кодируется в десятичной системе по 4 бита на десятичный разряд.

+3 Ответить

НЛО прилетело и опубликовало эту надпись здесь

lubezniy
19 сен 2020 в 08:07

Смотря для каких задач. Если надо потом вывести двухцифровое значение, например, на пару семисегментных индикаторов, которые подцеплены к GPIO через дешифраторы, то с такой кодировкой вывод делается по сути одной командой. Но да, вряд ли в применениях x86 (если не считать очень специфических) это очень полезно.

+1 Ответить

Vitalley
19 сен 2020 в 12:15

Ещё в 8080 была команда DAA команда для двоично-десятичной коррекции и её использовали для вывода чисел.

0 Ответить

lubezniy
19 сен 2020 в 12:30

Да, у Z80 она так же называлась. В принципе допускаю, что и в сугубо компьютерных приложениях с ней могло было быть удобно преобразовывать небольшие числа в разряды для вывода на экран или каких-то расчётов, связанных именно с десятичными разрядами.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Antervis
20 сен 2020 в 05:03

> Имхо, жуткий формат, когда число кодируется в десятичной системе по 4 бита на десятичный разряд.

Вот для конвертаций числа <-> строки, которые прикладной софт делает постоянно, такой формат очень удобен. И инструкции преобразования bcd<->binary тут бы помогли.

0 Ответить

netch80
21 сен 2020 в 19:56

Команды типа AAA, DAA… — не помогли бы. Эти команды хороши только для варианта операций непосредственно с десятичными числами в BCD без преобразования. И то, достаточно дёшево реализуются только сложения и вычитания. Умножение уже сложно, деление — кошмарик, дальше можно и не вспоминать.
Реально для сколь-нибудь сложных операций лучше сразу перевести в двоичное и работать в нём. Причём и для этого перевода не нужны BCD команды, умножения работают лучше.
Это касается и десятичной арифметики с плавающей точкой (финансовые расчёты):

использовать десятичный порядок при основной двоичной арифметике проще и надёжнее. Поэтому всякие DAA и выкинули наконец из x86-64, а в ARM и другие новые архитектуры даже не вводили. Кто их вынужден тянуть по легаси (как SystemZ), как-то дотягивают, но новый софт просят на них не делать.

+1 Ответить

Antervis
21 сен 2020 в 20:33

> Эти команды хороши только для варианта операций непосредственно с десятичными числами в BCD без преобразования

да это и не нужно. Профит может быть от самого преобразования bcd<->bin и именно для преобразования чисел в строки.

▸ Если интересно подробнее

0 Ответить

netch80
21 сен 2020 в 20:56

> Профит может быть от самого преобразования bcd<->bin и именно для преобразования чисел в строки.

Вот вам поступило, например, число стандартного размера int32 (до 2 миллиардов с хвостом), как BCD операции помогут перевести его в десятичный формат? А если int64?

На командах типа DAS этого не сделаешь. Нужно полноценную конверсию с делением.

> a idiv очень противная операция.

Мнэээ… простите, какой idiv? Компиляторы уже много лет заменяют деление на константу на обратное умножение. Вот я сходил на godbolt — скопирую результаты:

```
unsigned d100(unsigned num) {
    return num / 100;
}
```

```
d100(unsigned int):
        mov     eax, edi
        imul    rax, rax, 1374389535
```

```
shr     rax, 37
ret
```

Работает в разы быстрее (условно говоря, 3 такта вместо 70).

> В то же время вычислить длину и распечатать bcd — тривиальная задача. Длина через clz без if'ов, печать — цикл со сдвигом на 4 бита.

Это понятно. Вопрос в другом: BCD резко теряет эффективность, когда вам надо сделать больше, условно, 3 арифметических операций с числами. От этого уровня выгоднее потратиться на преобразование в двоичное представление, операции в нём и обратную конвертацию.
А где все операции короче такого? Это калькулятор и Excel (который в базовой функциональности — калькулятор на таблице).

> Кстати, сконвертировать в bcd вручную векторизацией получается намного медленнее, я пробовал.

Медленнее чего? Надо сравнивать не с другой конвертацией, а полный цикл конвертации, вычислений и обратной конвертации.

+1 Ответить

Antervis
21 сен 2020 в 21:14

> Вот вам поступило, например, число стандартного размера int32 (до 2 миллиардов с хвостом), как BCD операции помогут перевести его в десятичный формат? А если int64?

удвоением ширины регистра? На худой конец для больших чисел можно разбивать число на 2, всё равно быстрее, особенно учитывая что печатаются чаще маленькие числа

> Работает в разы быстрее (условно говоря, 3 такта вместо 70).

скорее 5 против 42, но да, эту особенность я забыл.

> Это понятно. Вопрос в другом: BCD резко теряет эффективность, когда вам надо сделать больше, условно, 3 арифметических операций с числами. От этого уровня выгоднее потратиться на преобразование в двоичное представление, операции в нём и обратную конвертацию.

так никто и не говорит об арифметических операциях в bcd формате, боже упаси. Только конвертация в строки и обратно.

> Медленнее чего? Надо сравнивать не с другой конвертацией, а полный цикл конвертации, вычислений и обратной конвертации.

медленнее описанной мной выше референсной реализации.

0 Ответить

netch80
21 сен 2020 в 23:33

> удвоением ширины регистра?

Так и в 32 битах таких операций нет. Ну, в x86.
Вот в System/Z есть как наследие S/360. Но — не рекомендуется.

> так никто и не говорит об арифметических операциях в bcd формате, боже упаси. Только конвертация в строки и обратно.

Которая всё-таки неплохо делается двоичными операциями — лучше, чем если бы реализовывалась в BCD.

> медленнее описанной мной выше референсной реализации.

Ну, по-моему, там вообще в рамках одного конвертируемого числа векторизовать тупо нечего. Поэтому неудивительно.

0 Ответить

Antervis
22 сен 2020 в 02:22

> Так и в 32 битах таких операций нет. Ну, в x86.

в смысле мы всегда можем положить число в регистр вдвое шире а потом преобразовать, и с переполнением никогда не столкнемся.

> Которая всё-таки неплохо делается двоичными операциями — лучше, чем если бы реализовывалась в BCD.

да точно не лучше. Вот смотрите, есть у вас какое-нибудь число а-ля 119. Если преобразовать его в bcd, будет 0x119. Для перекладывания в строку внутри цикла достаточно and 0xf, add '0', shr 4, трех однотактовых операций. В бинарном варианте — полюбуйтесь сами, на методы count_digits и format_decimal_result, еще можно глянуть в листинг лукап табличек.

> Ну, по-моему, там вообще в рамках одного конвертируемого числа векторизовать тупо нечего. Поэтому неудивительно.

попытайтесь реализовать и оптимизировать конвертацию в bcd-формат, сразу найдете простор для векторизации

-1 Ответить

netch80
22 сен 2020 в 07:46

> Если преобразовать его в bcd, будет 0x119. Для перекладывания в строку внутри цикла достаточно and 0xf, add '0', shr 4, трех однотактовых операций. В

Я про арифметику говорил, а не про текстовую конвертацию.
Во сколько раз на BCD дороже хотя бы сложение, вычитание, умножение, деление? Даже с аппаратной поддержкой?
А если перейти к плавучке?

> полюбуйтесь сами,

Полюбовался. Ничего фантастического. Деление и умножение в цикле с некоторым ускорением.

> попытайтесь реализовать и оптимизировать конвертацию в bcd-формат, сразу найдете простор для векторизации

Для целых фиксированного размера, типа int32 — не окупится, разгон дороже. Для более длинных — нереально.

+2 Ответить

Antervis
22 сен 2020 в 11:34

> Я про арифметику говорил, а не про текстовую конвертацию.

А зачем вы вообще говорите про арифметику? Я с самого начала треда, уже три раза как, написал что нужна именно сама конвертация, а не bcd арифметика

> Полюбовался. Ничего фантастического. Деление и умножение в цикле с некоторым ускорением.

ну с таким же успехом можно сказать что криптография без аппаратного ускорения тоже «умножение да сдвиги в цикле с некоторым ускорением». И всякая архивация/декодинг. Вот одну конкретную инструкцию преобразования bin->bcd могли бы и добавить/оставить.

0 Ответить

netch80
22 сен 2020 в 12:40

> А зачем вы вообще говорите про арифметику?

Затем, что конвертация это не главная цель, а трижды вспомогательная и занимает очень малую часть затрат в обычных задачах.
И то, основные проблемы в конвертации плавучих чисел — которые этим не решатся.

> Вот одну конкретную инструкцию преобразования bin->bcd могли бы и добавить/оставить.

Просто не нужно (на фоне прочих проблем).

+1 Ответить

aamonster
18 сен 2020 в 20:02

"Ну вы уже наверное знаете, что Современные iPad практически не уступают 15-дюймовым MacBook Pro с процессорами Core i7 и Core i9" – э… Не знаю такого. По просочившимся сведениям – мак мини на эппл силикон чуть слабее, чем на i3. Ждём более шустрых (и более горячих) армов.

+8 Ответить

NetBUG
19 сен 2020 в 15:58

Тут начинается task-oriented сравнение. И воспоминания о том, что ASIC может одну операцию (скажем, сжатие в JPEG, или вычисление каких-нибудь хешей) делать эффективнее GP CPU

0 Ответить

Antervis
20 сен 2020 в 04:03

> По просочившимся сведениям – мак мини на эппл силикон чуть слабее, чем на i3

это не мак мини, это developer kit, и поставленный в него A12Z по сути чип из 2018-ого года, который сейчас выдает в geekbench 5 1100/4700 очков, столько же, сколько i3-10300. При том что энергопотребление i3 больше *в 10 раз*.

Емнип ранний бенчмарк дев кита запускали в режиме эмуляции x86 кода и еще тогда аналитики были очень оптимистичны.

0 Ответить

aamonster
21 сен 2020 в 10:13

Ну да. Я и говорю – будем ждать более шустрых и горячих армов.

0 Ответить

justhabrauser
18 сен 2020 в 21:52

> Вы наверняка знаете, что мир процессоров разбит на два лагеря.
Сказал — как отрезал.

Хотя, в принципе, таки два:
— Цифровые vs аналоговые
— Скалярные vs векторные
— фон Нейман vs Гарвард
— CISC/RISC/MISC/VLIW… ой, тут четыре, не считается.
А так ровно два, да.

+18 Ответить

NIKOSV
20 сен 2020 в 06:50

Какая доля у MISC/VLIW на консьюмерском рынке?

0 Ответить

tyomitch
21 фев 2021 в 10:44

VLIW-процессоры стоят примерно в каждом модеме и в каждой камере.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

beeruser
19 сен 2020 в 01:18

> Одна и та же команда на x86 выполняется за 3 шага, а на ARM за 9, а ещё есть SIMD инструкции где ситуация ещё более грустная.

Какая команда? Какие шаги?
А что с SIMD инструкциями? Слышали про Neon, SVE?
Современные CISC процессоры имеют как раз таки более «CISC-овое» ядро чем старые (K6/Pentium Pro)

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

DGN
18 сен 2020 в 22:16

А что с расходом памяти? ARM прожорливее i386?

-5 Ответить

maxzhurkin
18 сен 2020 в 22:41

Процессоры не жрут память, они с ней работают

+14 Ответить

vanxant
19 сен 2020 в 17:05

Да, и требуют выравнивания данных в этой самой памяти. x86 исторически жрёт невыровненные адреса (наследство 8088). А вот если говорить про арм — то данные должны быть выровнены на 8 байт минимум. Если вы полезете в память по невыровненному адресу, процессор кинет аппаратное исключение и пошлёт вас чинить компилятор. Впрочем, на большинстве реальных арм-камней стоит аппаратный обработчик этого аппаратного исключения, но он не всесилен — в отличие от x86 не может обработать ситуацию на границе страниц ДРАМ (8кб).
В реальности вам редко нужны 64-битные целые (указатели да, хотя оверкилл, но вас не спрашивают; double когда как; а вот именно целые скорее всего нет, обычно хватает 32-битного int или вообще char/bool). Но компилятор по умолчанию всё равно выделяет им по 8 байт, чтобы не нарваться на исключение. Так что технически одна и та же программа под армом может жрать больше памяти, чем под x86. А разбор внешних упакованных структур (например IP-пакетов, форматов файлов и пр.) превращается в утомительное жонглирование байтами в регистрах. Хотя компилятор это спрячет «под капот», длина кода будет сильно выше.

-1 Ответить

DistortNeo
19 сен 2020 в 19:51

Не знаю как в ARM, а вот в x86-64 есть фича — зануление верхних 32 бит 64-битного регистра при использовании 32-битных операндов. Это позволяет сократить место на хранение индексов.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

CoolCmd
19 сен 2020 в 22:02

> А вот если говорить про арм — то данные должны быть выровнены на 8 байт минимум.

компилятор C++ для x86 использует по-умолчанию 'natural alignment': выравнивание равно размеру данных. и гугл говорит, что на arm та же фигня. откуда инфа про 8 байтов?

+3 Ответить

tyomitch
19 сен 2020 в 23:57

На смежную тему: в многопроцессорных системах выделение переменным разных строк кэша (т.е. выравнивание на >64 байта) может ускорять программу, предотвращая cache

eviction при работе разных процессоров с соседними переменными.

0 Ответить

vanxant
20 сен 2020 в 01:43

Ну а чего спорить то в эпоху онлайн компиляторов.
Возьмите godbolt и посмотрите, во что скомпилится следующая структура на C

```
#include "stdio.h"
struct Bad_Align
{
bool b;
char c;
short s;
int i;
float f;
double d;
void* p;
}  ba;
int main() {
printf("%d",sizeof(ba));
return 0;
}
```

+1 Ответить

CoolCmd
20 сен 2020 в 11:24

тыц
и в arm 32/64, и в интеле 32/64 размер структуры 32 байта. выравнивание полей, как я сказал выше — natural alignment.
сама структура выравнивается по размеру самого длинного типа в структуре.
так откуда инфа про:

> данные должны быть выровнены на 8 байт минимум

+2 Ответить

НЛО прилетело и опубликовало эту надпись здесь

edo1h
20 сен 2020 в 05:12

и не только на arm. так что избегать невыровненных чтений на C — нормальная практика (а на более высокоуровневых языках, даже если работа с указателями доступна, её следует избегать)

0 Ответить

DistortNeo
20 сен 2020 в 11:15

SIMD-операции на x86 тоже раньше бросали исключения.

0 Ответить

netch80
21 сен 2020 в 20:01

Ну почему же, есть ещё вопрос плотности системы команд. Вот я приводил варианты компиляции одной и той же программы в одинаковых режимах под разные архитектуры. Можно заметить, x86-64 и ARM/64 вровень, но есть и сильно более прожорливые (MIPS'ам не повезло).
Эту таблицу можно дополнить: я чуть более современный вариант той же программы проверил на gcc10 — 420K для x86-64 и 320K для RV64G (RISC-V 64-битный с разрешением "сжатых" команд — сразу сократило объём на четверть).

+1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

thealfest
19 сен 2020 в 09:17

Который все равно проигрывает по размеру коду x86\x64

0 Ответить

beeruser
19 сен 2020 в 14:55

Это не так. x86-64 гораздо более рыхлый чем x86.

Aarch64 код не только плотнее, но и обычно требует меньше инструкций чем x86-64.

+2 Ответить

edo1h
19 сен 2020 в 16:29

> x86-64 гораздо более рыхлый чем x86

131 / 126 ≈ 1.04

+3 Ответить

tyomitch
19 сен 2020 в 21:40

Я насчитал соотношение размеров 97/101/106/107 для AArch64/ARMv7A/i386/x86_64 соответственно.

0 Ответить

CoolCmd
19 сен 2020 в 22:07

у этого llvm исходный код одинаковый для всех процессоров?

0 Ответить

Justlexa
19 сен 2020 в 00:59

Некстати вопрос из любопытства про декодеры инструкций тех же ARM или IA64:
с чем связан выбор тех или иных битов в инструкции для указания imm, SIB?
Например в arm thumb инструкция MOVT кодируется так:

11110 i 101100 imm4 0 imm3 Rd imm8
и imm побитово собирается потом в порядке *imm4, i, imm3, imm8*
Для IA64 вариативность ещё шире.
Какие особенности декодера инструкций (или чего-то ещё) это диктуют?
x86-64 конечно тоже завёз таких отдельных битов с префиксами Rex.XXX, но тут всё же дело в совместимости с x86.

0 Ответить

yleo
19 сен 2020 в 02:36

IA64 это на самом деле Итаниум, там VLIW.

Если же говорить про IA32, то основной жупел случился из-за многократного "улучшения и расширения" набора команд с сохранением совместимости. Причем команды изначально были переменной длины, начиная с однобайтовых (что привело к неэффективному и запутанному использованию пространства кодов). Короче, думали не про декодер инструкций, а про совместимость и маркетинг.

0 Ответить

beeruser
19 сен 2020 в 02:40

В системе команд ARM не было 16-битных иммедиейтов.
Вот операнд и распихали в доступные места согласно карте инструкций.
Если посмотреть чисто регистровые инструкции, например MLS,
видно, что в этих битах (imm4/imm3) находятся регистры Rn, Ra.

Стараются ~~мух и котлеты~~ опкод и регистры+данные держать отдельно, и не перемещать поля потому как таким образом их проще читать и мультиплексировать.
ALU команды с непосредственными операндами так же содержат поля imm вместо одного из регистров, но размер операнда ограничен по понятным причинам.

imm8 в младших битах это традиционное место хранения непосредственного операнда и в ARM режиме.
iitd-plos.github.io/col718/ref/arm-instructionset.pdf
См. 4-2 / 4-10 (Operand2)

+1 Ответить

VaalKIA
19 сен 2020 в 06:36

> И если бы Intel лицензировал x86 за деньги другим людям, то вероятно Apple просто адаптировали свою текущую микроархитектуру под x86.

Эппл, купила компанию, которая разрабатывала PowerPC (силами именно этих разработчиков

она сделала рывок и лучшие APM процы), вполне могла бы возобновить разработки, но зачем ей PowerPC или x86, если она уже по уши в разработке APM?

+2 Ответить

LevOrdabesov
19 сен 2020 в 11:26

Учитывая, что делать RISC'и в целом проще (грубо говоря, их может клепать слабоквалифицированный дядя Вася в сарае), есть большая вероятность, что будущие неизбежные заплатки «очень-серьёзная-уязвимость-минус-20-процентов-производительности» сведут на нет все текущие преимущества архитектуры.

-2 Ответить

leossnet
19 сен 2020 в 14:44

Мне кажется, что когда говорят про Apple, то это не про технологии, а про бабки. Когда говорят, что производительность телефонов/планшетов Apple на ARM почти не уступает производительности компьютеров Apple на Core i7/i9, то это всего лишь говорит об избыточной мощности архитекторы x86 для программного обеспечения, которое работает на оборудовании Apple.

Отсюда для бизнесменов из Apple следует достаточно очевидный вывод, сформулированный по результатам проведенного функционально-стоимостного анализа (ну или как там это сегодня называется), что в их продукцию можно легко впихнуть более дешевый процессор без потери потребительских свойств оборудования для конечного пользователя (то есть для фанатов Apple).

Вот только нужно предварительно провести маркетинговые мероприятия по раскрутке новой крутой архитекторы собственных процессоров, чтобы после снижения себестоимости фанаты продолжили бы покупать продукцию Apple по старой цене. В результате и фанаты довольны, и у Apple растет прибыль. Как говорится «ничего личного, это просто бизнес».

+4 Ответить

Gumanoid
19 сен 2020 в 16:20

Вот для примера некоторые «RISC» инструкции ARMv8:
UZP1 — Unzip vectors
TBL — Table vector Lookup
FRSQRTE — Floating-point Reciprocal Square Root Estimate
SHA512H — SHA512 Hash update part 1
AESD — AES single round decryption

+3 Ответить

beeruser
19 сен 2020 в 20:21

Зачем вы использовали кавычки?
По всем канонам это обычные RISC операции регистр-регистр.
Они элементарно реализуются в железе.
Более того, они имеются почти во всех RISC процессорах.
Большая часть это просто шафл/регистровый лукап.
Реализуется комбинаторным блоком.
FRSQRTE чуть посложнее, но это просто табличный лукап.

Забавно, что в ARM есть load-op инструкции, свойственные CISC, но вы не написали ни одну.

-1 Ответить

Gumanoid
19 сен 2020 в 22:07

Ну это тонкий момент что считать RISC, а что CISC. Если исходить из аббревиатуры, то у риска набор команд ограничен, поэтому всякие квадратные корни и хэши выглядят странно.

+3 Ответить

beeruser
20 сен 2020 в 13:08

> Ну это тонкий момент что считать RISC, а что CISC.

Есть вполне чёткие критерии RISC ISA.

> Если исходить из аббревиатуры, то у риска набор команд ограничен, поэтому всякие квадратные корни и хэши выглядят странно.

Выглядит для вас странно, а на самом деле совершенно естественно.
Разработчики RISC архитектур так же с вами совершенно не согласны.
Вы запрещаете им добавлять новые команды в ISA? Так?

Смысл RISC не в общем количестве команд, а в:
1) Избегании *лишних* команд, которые не используются компиляторами или без которых можно обойтись.
2) Разделении команд на вычислительные и команды для работы с памятью.
3) Использовании простых команд, которые можно реализовать в «железе» вместо микрокода.

Как я и написал, то что для вас кажется сложной командой, на самом деле простая.
Вычисление корня или шафлы по-вашему должны на RISC процессорах тормозить?

-1 Ответить

creker
20 сен 2020 в 14:53

> Есть вполне чёткие критерии RISC ISA.

Для вас может есть. Но во всем остальном мире четких критериев, которых все придерживаются, нет. Тем более таких, чтобы в них хорошо помещался APM современный. Иначе бы этот спор, что RISC, а что нет, не существовал банально.

> Использовании простых команд, которые можно реализовать в «железе» вместо микрокода.

Довольно странный критерий. Микрокод явно используют не потому что невозможно что-то в железе реализовать. Его использовать, потому что что-то выгоднее в железе делать, а что-то нет. При желании все можно было засунуть в железо. ARM нынче содержит инструкции вроде AES и CRC, которые явно не простые, но в железе, как и в x86. Это прямо противоречит RISC подходу и заимствование у CISC, когда в ISA добавляются инструкции, которые нужны реальным приложениям. Не всегда, но довольно часто. В противоречие первому пункту вашему, без них тоже можно было обойтись. Не говоря уже о всяких микроопсах, микроопс фьюзинге и прочих CISC трюках, призванных ускорить выполнение сложных инструкций.

> Вычисление корня или шафлы по-вашему должны на RISC процессорах тормозить?

А зачем в ISA то, что может сделать код самостоятельно? Если мы придерживаемся RISC идеологии, то это все должен делать софт. Без инструкции можно обойтись? Можно. Но зачем-то даже в RISC-V похоже добавили это, хоть и в качестве расширения, а не части основного ISA. В арме так вообще стало обязательной частью, как и AES с CRC.

+2 Ответить

beeruser
21 сен 2020 в 19:40

> Тем более таких, чтобы в них хорошо помещался APM современный.

ARM руководствуется принципами эффективности, а не архитектурной частоты.

> Иначе бы этот спор, что RISC, а что нет, не существовал банально.

О чём спор? Вы показываете красный носок и говорите что он зелёный?
Если вас пугают крипто-инструкции, то уж три первые вами упомянутые инструкции это

100% RISC инструкции без каких-либо сомнений.

Что же делает SHA512H? Читает 3 регистра, выполняет некую операцию и записывает обратно в регистр. У неё нет ни состояния, ни чего либо ещё. Ничем не отличается от какого-нибудь FMAD, только проще. Т.е. умножение и сложение — RISC, а эта операция не RISC? Л — Логика.

Что характерно, вы так и не осилили назвать CISC инструкции ARMv8, а они есть.

> А зачем в ISA то, что может сделать код самостоятельно?

Ответ очень прост — производительность и объём кода.
По-вашему получается и SIMD не нужен, ведь всё можно сделать на скалярном ALU :)

> Если мы придерживаемся RISC идеологии, то это все должен делать софт

Это где такое написано? Сами выдумали?

> Это прямо противоречит RISC подходу и заимствование у CISC, когда в ISA добавляются инструкции, которые нужны реальным приложениям.

То что вы говорите не имеет никакого смысла.
RISC как раз про эффективность. И если реальные приложения могут получить выгоду от конкретных инструкции, они там просто обязаны быть.

> Не говоря уже о всяких микроопсах, микроопс фьюзинге и прочих CISC трюках, призванных ускорить выполнение сложных инструкций.

Это не CISC трюки. Это микроархитектурные трюки. Нужно понимать разницу.
Простой ADD со сдвигом может разбиваться на 2 инструкции на сложном ARM процессоре и выполнять за 1 такт на простейшем микроконтроллере.
Не потому что инструкция какая-то CISC-овая. Просто она не успеет отработать на (целевой) высокой частоте из-за задержек.

> Но зачем-то даже в RISC-V похоже добавили это, хоть и в качестве расширения, а не части основного ISA.

Потому что вы себе придумали какую-то альтернативную трактовку RISC и возмущаетесь тому что говорю я или тому что делают Дэвид Паттерсон и все остальные архитекторы MIPS. Power, ARM.

Это базовые возможности для современных процессоров.

Конкретно криптоакселератор я бы предпочёл иметь сопроцессором,
но уж возмущаться SIMD и аппроксимации квадратного корня это странно, мягко говоря.

0 Ответить

DX168B
19 сен 2020 в 16:39

Для того, чтобы ARM начал нормально отвоёвывать рынок серверов и рабочих станций, нужно разработать стандартную архитектуру компьютера, как это было сделано в компании IBM в свое время (IBM PC). Именно благодаря общепринятому стандарту мы сейчас имеем универсальную платформу и ПО. А пока этого нет, ARMу будет тяжко в этом сегменте.

+2 Ответить

BD9
19 сен 2020 в 22:51

Школота, мля.
Школота здесь, школота там.

ARM и так импользует «стандартную архитектуру компьютера, как это было сделано в компании IBM в свое время (IBM PC)» — PCI-E, память DDR3-4 и т.д.

По статье:
Грамматика никакая.

> Даже вы можете начать производить свои процессоры, купив лицензию. А вот производить процессоры на x86 не может никто кроме синей и красной компании.

— есть VIA и несколько СП с китайцами.

> И если бы Intel лицензировал x86 за деньги другим людям, то вероятно Apple просто адаптировали свою текущую микроархитектуру под x86. Но так как они не могут этого сделать, они решили просто перейти на ARM. Проблема для нас с микроархитектурой в том, что она коммерческая тайна. И мы про нее ничего не знаем.

— защита x86 патентами кончилась по истечении сроков.

> в мире x86 творится монополия

— такого никогда не было.
.
.
После покупки ARM Нвидией Яблоко может убежать на другие наборы команд.

+1 Ответить

geher
20 сен 2020 в 13:02

> защита x86 патентами кончилась по истечении сроков

Но новые наборы инструкций и некоторые нюансы архитектуры новых процессоров таки защищены новыми патентами, что несколько осложняет создание актуального и, что важно, мало-мальски совместимого x86-64.
Совместимость уровня того же четвертого пня уже почти никому не нужна.
А развивать новую экосистему вокруг того же x86, только немного другого (а потому в некоторых вопросах несовместимого) — трудно будет обосновать потребителям, зачем им переходить туда из привычного x86-64 от AMD-Intel.

> | в мире x86 творится монополия
> — такого никогда не было.

По крайней мере когда появился первый 8086 — была абсолютная монополия — 100%, хотя и недолго. :)
Впрочем, Intel вроде достаточно долго держал больше половины рынка x86, что вроде как достаточно, чтобы считать его монополией.

0 Ответить

VaalKIA
20 сен 2020 в 17:34

> Но новые наборы инструкций и некоторые нюансы архитектуры новых процессоров таки защищены новыми патентами

Если вы помните как Интел топил за Итаниниум, то должны помнить и то, что Интел заявляла, что не будет 64 бит x86. Поэтому теперь Интел лицензирует 64битное расширение комманд у АМД. И по сути оно amd64, а не x86-64. А уже помимо этого существуют всякие sse и прочее. Просто напишите, какой сет вы считате необходимым и его нельзя лицензировать.

0 Ответить

geher
20 сен 2020 в 20:29

Насколько я наслышан, Intel и AMD сильно повязаны этой архитектурой (не может AMD без технологий Intel, а с некоторых пор и Intel без технологий AMD) и антимонопольным законодательством (не может Intel без AMD), а потому достаточно свободно лицензируют друг другу всякие технологии.
Тем, кто вне этой пары, вроде все уже не так просто, судя по некоторым косвенным данным.

> И по сути оно amd64, а не x86-64

И как её только не обзывали, эту архитектуру.

0 Ответить

lwanowsky
21 сен 2020 в 00:36

В 2015г. AMD лицензировала китайцам ядро Ryzen первого поколения; но понятное дело, китайцы не могут получать последующие разработки AMD для актуализации своего микропроцессора, и тот идет теперь своим китайским путем.
А когда-то кто только x86-микропроцессоры не выпускал: Texas Instrumentis, Cyrix, UMC, IBM, NexGen, Transmeta (Crusoe), IDT (WinChip), Rise Technology (mP6), Chips&Technologies и др.

0 Ответить

DX168B
19 фев 2021 в 19:08

Пора бы уже отличать «архитектуру процессора» от «архитектуры компьютера».
Архитектура всем известного компьютера называется IBM PC. Это целая экосистема и свод стандартов, позволяющих без излишних танцев с бубном запустить софт на IBM PC совместимых компьютерах любых производителей. То же самое касается и дополнительного железа. А вот про ARM такого не скажешь. Нет для ARM единого стандарта построения компьютера, такого как IBM PC. Не изобрели еще.
Еще раз повторяю, не процессора, а компьютера. Процессор — это часть компьютера/аппаратной платформы.

0 Ответить

DrPass
19 фев 2021 в 21:52

> Архитектура всем известного компьютера называется IBM PC

Интересно, с чего бы архитектура всем известного компьютера называется IBM PC, если IBM PC был выпущен четыре десятилетия назад, и с современным компьютером не совместим ни аппаратно, ни архитектурно, ни программно?

> Нет для ARM единого стандарта построения компьютера

А что такое «стандарт построения компьютера»? ARM-системы используют абсолютно те же спецификации, что и системы на базе x64.

+1 Ответить

iproger
20 сен 2020 в 05:34

Я думаю имелись ввиду обычные рабочие станции. Грубо говоря, чтобы по «arm pc buy» можно было сразу купить себе компьютер и начать на нем разрабатывать.
Raspberry pi и аналоги не могут быть заменой нормальному пк.

0 Ответить

DX168B
19 фев 2021 в 19:14

Да, именно это я и имел в виду. Если Apple возьмется за разработку стандарта, аналогичного стандартам IBM-PC, то у ARM будут все шансы потеснить x86 на рынке.

0 Ответить

v1000
19 сен 2020 в 23:07

С ARM процессорами в своё время была интересная путаница в процессорах. Особенно когда не указывали тип процессора. И получалось, что один и тот-же «процессор 1GHz» мог выдавать разную производительность. А покупателю оставалось удивляться, что его телефон или планшет тормозят. А получалось это из-за того, что в процессоре, к примеру, небыло инструкций сопроцессора, и команду, которую другой процессор мог выполнить за один такт — этот выполнял за три.

0 Ответить

redsh0927
20 сен 2020 в 11:59

«RISC» и «CISC» — чистейший маркетинг. Ну, может быть эти слова и имели какой-то технический смысл лет 40 назад, когда процессор A умел только плюсовать, но не умел умножать, а процессор B умел и плюсовать и умножать, но тратил на это 100 тактов. Сейчас же все процессоры выполняют одни и те же операции, имеют на борту математические и векторные сопроцессоры, всякое шифрование и выполняют суммарно по сотне умножений за такт. Никакого риск и циск уже сто лет не существует. BTW, «трансляция команд x86», про которую орут из каждого утюга — такой же бред. ЛЮБОЙ сколько-нибудь серьёзный цпу использует микрокод и микрооперации! x86 даже не первый! Существенная разница в системах команд между арм и x86 в том, что x86 умеет адресовать память из любой команды, а арм — использует RMW-подход, зато имеет 13 регистров вместо 7. И это в принципе неплохо. Но главное — x86 имеет сложную систему кодирования команд переменной длины, а арм обходится 2 или 4-байтовыми командами. Как по мне, эта фиксированная длина команды — полнейшее говнище. Несколько упрощая реализацию процессора, она превращает код в полнейшее месиво. Простая загрузка константы в регистр или считывание значения по imm-адресу разворачивается в целую простыню мусора. Все эти таблицы констант, пихаемые между функциями — вот настоящие костыли. «Эффективному RISC» приходится по 2 раза лезть в память, сперва считывать команду, затем доставать константу из таблицы для простейшего mov reg,imm32! Но зато всякие NOP и прочие операции на одном регистре или вообще без аргументов занимают целое слово… И россказни про сложный декодер x86 команд совершенно

несостоятельны. Кэши занимают половину кристалла, ещё львиную долю исполнительные блоки, декодеры же команд с лупой искать надо…

+6 Ответить

VaalKIA
20 сен 2020 в 17:48

В целом всё верно, но на декодере заостряют внимание потому, что он должен поставить комманды во все конвееры для спекулятивных вычислений, так что он должен работать гораздо быстрей чем цепочка реально исполняемых комманд. И вот тут начинаются проблемы, потому что он по тем же топонормам делается и работает, и что бы он был не «такой как все», желательно бы его попроще или сделать специальные телодвижения, которы бы позволяли выбирать последующие комманды когда ещё текущие не декодированны. Эту проблему как раз решает фиксированная длинна комманды.

0 Ответить

creker
20 сен 2020 в 18:45

Как видно, вполне себе справляется. Более того, поспевает даже в случае SMT, когда, по идее, темпы фетчинга команд еще выше. Ходят слухи, что АМД и SMT4 может добавить.

+1 Ответить

netch80
21 сен 2020 в 20:24

> Эту проблему как раз решает фиксированная длинна комманды.

У x86 бóльшая проблема этого декодера не в переменной длине, а в количестве промежуточных шагов, которые необходимы, чтобы определить реальную длину. Выбрали первый байт. Декодировали. Нашли префикс. Идём к следующему байту. Однобайтный код операции, ОК. Узнали, что за ним идёт mod-reg-r/m, который надо декодировать для получения аргументов. Выбрали. Декодировали. Узнали, что за ним идёт SIB, который надо разобрать. Выбрали. Декодировали… Да, это параллелится. Но дорого — на каждый шаблон свой детектор на несколько байт. Не зря говорят "выравнивайте точки перехода на 16 байт", потому что запускать весь декодер с любой позиции — дорого… делают, конечно, но медленнее.

Теперь сравним с System/Z. Выбрали два первых байта, в них есть два бита в конкретном месте: 00 — 2 байта длины, 01 и 10 — 4, 11 — 6. Декодер проще на пару порядков, не надо уметь быстро разбирать очень сложные конструкции.
То же для RISC-V, но там префикс длины с бóльшим количеством вариантов (считаем, почти любым), и тоже декодировать легко и просто.

0 Ответить

edo1h
21 сен 2020 в 22:59

я думаю, что если бы это заметно влияло на производительность, то уже давно бы сделали альтернативные машинные коды для тех же самых ассмеблерных инструкций.

+1 Ответить

netch80
21 сен 2020 в 23:29

> то уже давно бы сделали альтернативные машинные коды для тех же самых ассмеблерных инструкций.

Дублирующие коды для всех основных команд?
Если просто дублирование, то сразу каша — какие лучше и почему, при том, что уже наработаны гигабайты скомпилированного кода, который вряд ли будут менять: старый код надо продолжать поддерживать. То есть это будет уже два декодера, суммарная сложность только вырастет за счёт таки используемого старого варианта.

Полная перестройка при смене разрядности?
Вот тут — надо учитывать историю.
Выпустив 8086, пытаются опередить весь мир, сделав iAPX432. Не взлетает. В режиме судорожной затычки выпускают 286. Плохо взлетает, выпускают 386. Это уже на что-то нормальное похоже, но сделано в условиях, когда весь пар ушёл в свисток и надо запускаться на оставшихся копейках. Времени и сил на перекраивать нет, делают просто пришлёпки к имеющемуся. Только запустившись, оно обрастает легасями и менять уже нельзя — никто не пойдёт опять переписывать весь код.
Проходит 15-17 лет. Ресурсы есть, запускается очередная пирамида Хеопса — IA-64 ("ну, теперь точно весь мир взуем"), и затычка Pentium4. Не взлетает. Пока ожидают у моря погоды, нищий AMD в состоянии исторического минимума проектирует 64-битку, опять ничего не перекраивая, а только делая чуть нашлёпок. Взлетает, тут же создавая легаси. Intel барахтается ещё пару лет и заимствует полученное. И опять перекраивать нормально уже некогда, успели наработать много готового кода.

Это не только в формате команд. Вот, например, 32-битные операции над регистрами в 64-битном режиме чистят старшие половины. А 8- и 16-битные операции — не чистят. А почему? И снова легаси — они уже не чистят биты 31...16, значит зависимость по данным сохраняется, зачем тогда чистить 63...32?
Воображаю, какими словами Intel сейчас клянёт разработчиков i386 и IA-64, но "был бы я таким умным, как моя жена завтра"...

А ARM при переходе к 64 битам выбросил старые неадекватные заморочки и такой проблемой не страдает. Штош, они оказались умнее строителей пирамид.

0 Ответить

edo1h
22 сен 2020 в 01:20

> Дублирующие коды для всех основных команд?

что-то вроде thumb в arm (вернее, anti-thumb, пожалуй, за компактностью гнаться не надо). некий режим процессора, в котором те же самые мнемоники кодируются иначе — логично, удобно для декодера.

давай оно хоть пару процентов производительности — код для него быстро появился бы, благо в gcc/llvm приделать вообще ничего не стоит.

но я думаю, что выигрыша бы не было. могу ошибаться.

> Если просто дублирование, то сразу каша — какие лучше и почему, при том, что уже наработаны гигабайты скомпилированного кода, который вряд ли будут менять: старый код надо продолжать поддерживать

ну arm как-то живёт с несколькими системами команд

> То есть это будет уже два декодера, суммарная сложность только вырастет за счёт таки используемого старого варианта.

так вы говорите, что новый декодер может оказаться заметно проще (и притом производительнее). x64 приделали, 100500 вариантов simd приделали, а ещё один простой декодер не приделают?

> Вот, например, 32-битные операции над регистрами в 64-битном режиме чистят старшие половины. А 8- и 16-битные операции — не чистят.

это уже все кодогенераторы умеют и на скорость не влияет.

0 Ответить

khajiit
21 сен 2020 в 23:47

> Выбрали первый байт. Декодировали. Нашли префикс

Так это прямое следствие лапшекода вместо ISA, тут ничего не попишешь, оно с рождения такое.
Был момент, когда можно было это исправить…

0 Ответить

redsh0927
22 сен 2020 в 07:47

> Выбрали первый байт. Декодировали. Нашли префикс. Идём к следующему байту. Однобайтный код операции, OK. Узнали, что за ним идёт mod-reg-r/m, который надо декодировать для получения аргументов. Выбрали. Декодировали. Узнали, что за ним идёт SIB, который надо разобрать. Выбрали. Декодировали…

Ну и что, никто не будет последовательно разбирать команду. Декодируют одновременно по всем основным шаблонам. Ну, несколько больше шаблонов, чем для всякого риска. Но не «на пару порядков» уж точно. Процессор и есть сложный, десятки миллиардов транзисторов уже. Значит и функциональность должна быть соответствующая, а не «здесь густо, здесь пусто». Зачем делать крутой набор команд и экономить копейки на кодировании? Чтобы банальное чтение слова по имм-адресу разворачивалось в просыню мусора? Сделать процессор можно один раз и потом тупо печатать, а с убогими примитивными командами придётся всё время мучаться при разработке софта!

0 Ответить

netch80
22 сен 2020 в 08:01

> Зачем делать крутой набор команд и экономить копейки на кодировании? Чтобы банальное чтение слова по имм-адресу разворачивалось в просыню мусора?

Этого я как раз не предлагал — я ратую за переменную длину, но с префиксом длины. Тут уже можно и вложить всё вплоть до загрузки сверхдлинной константы.

Но:

> Чтобы банальное чтение слова по имм-адресу разворачивалось в просыню мусора?

Как часто такая операция вообще нужна?

> а с убогими примитивными командами придётся всё время мучаться при разработке софта!

Мучаться придётся двум небольшим группам — 1) авторам компилятора — причём только той части, которая переводит в целевой код, 2) авторам ассемблерных переходников. Остальным пофиг.

Вот две последние разработки ISA — AArch64 и RISC-V. Обе разрабатывались командами грамотных инженеров с оптимизацией каждый под свою специфику, но под последние достижения отрасли. Видна масса одинаковых решений, например: общая схема RISC, операции с памятью отделены от операций с данными; нет слотов задержек; нет условного выполнения команд, так любимых в ARM/32… Много различного. Но для "банального чтения слова по имм-адресу" в обоих надо загрузить этот адрес (или близкий к нему) в регистр и затем только прочитать. Может, оно в стиле PDP-11 "MOV @#1234,@-(R2)" и не нужно?
Да, это отсылка к авторитету, но когда два заметно разных авторитета дают примерно одно и то же — это уже наводит на мысли.

> Процессор и есть сложный, десятки миллиардов транзисторов уже.

Это только высшие модели, и то десятки миллиардов это на толпу ядер, у которых раздельные декодеры. А минимальные версии могут укладываться и в сотню тысяч. Да,

без OoO, конвейеров, кэшей и пр., но будет работать.

> Декодируют одновременно по всем основным шаблонам.

Я это сказал. Всё равно дороже. И да, думаю, раз 100 таки точно есть усложнения.

0 Ответить

redsh0927
22 сен 2020 в 09:42

> Может, оно в стиле PDP-11 «MOV @#1234,@-(R2)» и не нужно?

В мелкоконтроллерах постоянно приходится лазить в I/O-space и аналог характерного «or dword ptr [0x12345678],0x80000001» разворачивается в 7 команд и портит пару регистров. Да и просто аналог «or eax,0x80000001» написать нельзя — нужно доставать 0x80000001 из памяти, но и к памяти обратиться напрямую нельзя — нужно загружать адрес. Всё это выглядит крайне костыльно, нечитабельно, уродливо. Сотню раз исплюёшься пока что-нибудь на ассемблере напишешь. И всё это ради фиксированной длины… Ну сделали бы хоть бы 4/8, уже как-то можно было жить!
RISC-V похож на ARM, что тут удивительного, как альтернативу арму и делали…

0 Ответить

netch80
22 сен 2020 в 12:44

> or dword ptr [0x12345678],0x80000001

Ну именно простые микроконтроллеры и сейчас та область, где CISC стиля PDP-11 с потомками был бы очень вкусен. Всякое OoO там не нужно, даже минимальный конвейер часто излишен, тактовой достаточно низкой...

> RISC-V похож на ARM, что тут удивительного, как альтернативу арму и делали…

Нет, он MIPS, из которого выкинули ошибки прежней реализации. Внутри RISC мира это фактически полная противоположность.

0 Ответить

BD9
20 сен 2020 в 22:51

> Существенная разница в системах команд между арм и x86 в том, что x86 умеет адресовать память из любой команды, а арм — использует RMW-подход, зато имеет 13 регистров вместо 7.

— 8086 и 8088 имели 14 16-битных регистров. Далее их количество постоянно росло.

+2 Ответить

DistortNeo
20 сен 2020 в 23:08

Не путайте общее количество регистров и регистры общего назначения.

У 8086 — 14 регистров, их них 8 — общего назначения, полноценных — 7, причём они ещё и не равноправные.

У ARM — 16 целочисленных регистров, из них 13 — равноправные регистры общего назначения.

0 Ответить

rcl
20 сен 2020 в 20:46

Два лагеря это CISC и RISC, а не x86 и ARM. Кроме того, после покупки ARM компанией NVIDIA, большинство здравомыслящих компаний начнут смотреть в сторону более открытых архитектур, таких, например, как RISC-V.

-1 Ответить

NeoCode
20 сен 2020 в 20:54

Мне бы интересно было рассмотреть и сравнить наборы команд. Помню, еще будучи студентом, по книге Зубкова восстановил таблицу команд x86 и обнаружил там недокументированную команду с кодом 0xF1, и затем уже в инете нашел что это некая ICEBP (INT01).
Ну и таблицу команд ARM в классическом виде было бы интересно посмотреть.

0 Ответить

sadko4u
21 апр в 15:13

Статья весьма безграмотная, примеры и аналогии притянуты за уши.
Видимо, автор сам ~~пороха не нюхал~~ на ассемблере под обе сравниваемые архитектуры толком и не программировал. Всё отличие RISC от CISC можно пояснить двумя отдельными пунктами:

1. Длина команды: у CISC она переменная, у RISC — постоянная. Это отчасти верно, но есть особенности, т.к. у тех же ARM и RISC-V есть возможность исполнять сокращённые в два раза по длине команды для увеличения плотности кода.
2. Наличие отдельных выделенных команд пересылки данных между памятью и регистрами в RISC, в то время как в CISC команды могут в качестве операнда использовать ячейку памяти.

То, что в RISC какой-то убогий набор инструкций — уже давно миф и результат неправильного перевода самой аббревиатуры RISC на русский язык. В современных ARM и MIPS даже SIMD есть (который, к слову, местами попродвинутее SSE и AVX будет), который сейчас является одним из ключевых способов оптимизации производительности обработки данных.

+1 Ответить

khajiit
22 апр в 14:53

1 — справедливо только если оба набора могут выполнятся одновременно в одном сегменте кода. Если для переключения набора требуется специальная инструкция, метка сегмента/страницы, запись в регистр, whatever еще — то это или безграмотность, или окажется что сумматор и умножитель использую разные системы команд, что неверно.

0 Ответить

kovserg
22 апр в 16:55

Так для arm thumb2 как раз может такое если старший байт <0xF4 16бит инструкция иначе 32бит.

A5.1 Thumb instruction set encoding

The Thumb instruction stream is a sequence of halfword-aligned halfwords. Each Thumb instruction is either a single 16-bit halfword in that stream, or a 32-bit instruction consisting of two consecutive halfwords in that stream.

If bits [15:11] of the halfword being decoded take any of the following values, the halfword is the first halfword of a 32-bit instruction:

- `0b11101`.
- `0b11110`.
- `0b11111`.

Otherwise, the halfword is a 16-bit instruction.

0 Ответить

khajiit
22 апр в 20:33

Хорошо, процессоры с поддержкой Thumb2 можете считать не совсем чистыми RISC (а кто-то обещал реализацию прямо-вот академической модели?).
При том, до изменения длины инструкции в 15 раз при обилии префиксов и необязательных байт отсюда все равно что до Луны пешком, регулярность структуры сохранена; так что даже если судить настолько строго, все равно к чистому RISC он на порядок ближе, чем к реальному CISC.

0 Ответить

edo1h
23 апр в 00:13

Ну thumb/thumb2 — отдельная история. И в arm64 аналога этому режиму нет (пока?).

+1 Ответить

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

uzumeti
21 час назад

Как материалы расширяются при охлаждении и почему это очень странно?

Простой 3 мин 20K

+30 35 44

valisak
4 часа назад

Как менялось наше представление о Большом взрыве

Средний 10 мин 3K

Ретроспектива

+21 11 5

5kull_h4ck3r
1 час назад

Обратный отсчёт пошёл: последний шанс поучаствовать во взломе года

1 мин 612

+20 0 0

edlost
23 часа назад

Press F, чтобы рассчитать использование облачных ресурсов

7 мин 1.2K

+18 8 6

Seleditor
14 часов назад

Китай ускоряет процесс импортозамещения. Проблемы есть, но давление США пока не смогло остановить КНР

4 мин 4.3K

+15 3 37

Kilor
7 часов назад

PostgreSQL Antipatterns: ходим по JSON-граблям

Простой 3 мин 2.1K

Кейс

+14 25 0

7 часов назад

30 лет МТС: принимаем поздравления и делимся опытом

8 мин 2.5K

Спецпроект

+14 3 18

Albert_Wesker
5 часов назад

Что такое гексагональная архитектура. Разделение бизнес-логики и инфраструктуры с помощью портов и адаптеров

Средний 13 мин 1.2K

Обзор Перевод

+13 19 2

SLY_G
3 часа назад

Как прошёл эксперимент с китайской лунной микрофермой

6 мин 1.7K

Перевод

+12 4 7

MAX1993M
1 час назад

KMP перешёл в stable. Что это значит?

Простой 4 мин 458

Обзор

+10 3 0

Показать еще

ИНФОРМАЦИЯ

Сайт	droider.ru
Дата регистрации	30 ноября 2010
Дата основания	4 апреля 2010
Численность	2–10 человек
Местоположение	Россия

БЛОГ НА ХАБРЕ

17 мар 2022 в 16:31

Как устроены файлы? Разбор

22K 21

9 фев 2022 в 17:42

Что такое и зачем нужен IPV6? Разбор

74K 75

12 ноя 2021 в 16:58

Чем хорош чип Google Tensor? Разбор

32K 13

28 окт 2021 в 16:13

Какими бывают дисплеи в ноутбуках? Разбор

33K 15

---

11 окт 2021 в 16:42

Как работают Android-приложения в Windows 11? Разбор

24K 25