Оглавление (содержание)

0. Предисловие. 4

1. Перед началом тестов. 7

2. Rosetta 2 и чем отличаются macOS для ARM и x86_64. 11

2.1. В чём суть Rosetta 2 и чем она отличается от эмулятора. 11

2.2. Ограничения Rosetta 2. AVX инструкции. 13

2.3. В чём сложность трансляции AVX-инструкций. 23

2.4. Какие ещё есть ограничения у Mac на Apple Silicon? 26

2.5. Зачем нужна Rosetta 2 для универсальных приложений? 31

2.6. Неочевидные проблемы с macOS на M1 и их причины. 39

2.7. Какие есть отличия между macOS для x86 и ARM? 41

2.8. Внешние видеокарты (eGPU) и USB4/Thunderbolt. 56

3. Тесты SSD накопителя Apple AP0256 и объединённой памяти на подложке с чипом M1. 60

4. Троттлинг-тесты. 78

5. Тесты производительности процессора. 108

5.1. Cinebench R23. 108

5.2. Cinebench R20. 112

5.3. Geekbench 5. 114

5.4. Blender 2.91.2 и Blender 2.92. 122

5.5. Скорость компиляции Swift проектов для Mac в XCode. 126

5.6. Скорость компиляции Obj-C + Swift проектов для iOS в XCode. 131

5.7. Скорость компиляции крупного проекта на Rust. 136

5.8. Сборка крупных проектов с множеством компонентов и зависимостей на C, C++, Java и Python. 141

5.9. Скорость компиляции проекта на C + Assembler. 145
5.10. Сравнительная таблица с результатами тестирования процессора и выводы. 150
6. Тесты производительности интегрированного графического ускорителя. 154
6.1. Geekbench 5 Compute (API OpenCL, Metal). 155
6.2. GFXBench (API Metal). 161
6.3. Unigine Valley (API OpenGL). 165
6.4. Unigine Heaven (API OpenGL). 168
6.5. Сравнительная таблица с результатами тестирования GPU. 170
7. Тестирование блоков аппаратного кодирования и декодирования видео. 171
7.1. А что заявлено по аппаратным блокам у Apple M1? 171
7.2. Тесты с кодеками H.265 и H.264 при помощи ffmpeg. 175
7.3. Тесты с кодеком AV1 при помощи dav1d. 181
7.4. Тесты декодирования 8K и 4K видео в YouTube. 186
7.5. Тесты 8K и 4K видео в VLC. 192
7.6. Выводы по декодированию и перекодированию видео..... 198
8. Тестирование автономности. 199
9. Какое ПО (не) работает на Mac на Apple Silicon? 211
9.1. Родное ПО под macOS с Apple Silicon. 211
9.2. ПО под macOS на Intel. 215
9.3. ПО под iOS / iPadOS. 221
9.4. Эмуляция других платформ. 229
9.5. Виртуализация. Docker. 230
9.6. Приложения для Windows. 232

10. Тестирование игр. Методика и результаты. 250
10.1. Методика тестирования. 250
10.2. Asphalt 9. 257
10.3. Genshin Impact. 258
10.4. Shadow of the Tomb Raider (2018). 259
10.5. Rise of the Tomb Raider (2015). 262
10.6. Tomb Raider (2013). 263
10.7. Сравнительная таблица и выводы по играм. 264
11. Камера. 265
12. Wi-Fi. 267
13. Субъективные впечатления и выводы. 270

0. Предисловие.

За предоставленный на обзор аппарат выражаю благодарность интернет-магазину Цифрус. Переходите на их сайт и приобретайте любые товары по промокоду АйКакПросто и получайте за это небольшой подарок.

Сразу предупрежу, что статья вышла настолько огромной, что теперь тянет на небольшую книгу. Материала в ней вышло больше, чем на 260 страниц A4 (разумеется, это всё не сплошной текст, а ещё и более 250 скриншотов, 9 видео и 7 таблиц). Её не удалось вместить на нашем сайте на одной странице, поэтому мы решили просто прикрепить файл со статьёй для загрузки с сайта. Комментарии оставляйте на сайте, я их все тут читаю.

Ну что тут сказать? Это первый MacBook у меня на тесте и первый ноутбук на ARM64 (aarch64) процессоре, который я тестирую в принципе.

Был ли у меня опыт работы с macOS? Разумеется, да.

Был ли у меня опыт работы с ПК на ARM? Да, у меня дома работает одноплатный микрокомпьютер Raspberry Pi 4 с 4 ГБ оперативной памяти под управлением Ubuntu 20.04, который я для своих задач использую без графического интерфейса (он мне не требуется). Если наберётся доброе количество желающих в комментариях, я выпущу на него отдельный обзор на сайте с разбором его нюансов, коих я приметил дофига и больше.

Но что же такого специфичного в Macbook Pro 13 2020 на базе нового процессора Apple M1? Начнём с того, что это ноутбук с процессором на базе архитектуры ARMv8 (она же ARM64 или aarch64) с полноценной десктопной операционной системой macOS 11 Big Sur. В отличие от Manjaro, Ubuntu и других Linux-дистрибутивов под тот же Raspberry Pi 4, macOS в предыдущих своих версиях не имела поддержки архитектуры ARM и потому нативного софта под ARM на macOS ещё мало. А вот у популярных дистрибутивов Linux в репозиториях полно собранного/скомпилированного софта не только под x86_64, но и для aarch64. Многие разработчики ПО под macOS ещё не успели переписать свои приложения для нативной работы на этой архитектуре, но в будущем, определённо, число нативного софта для Macbook на Apple Silicon возрастёт. Вот только я пишу обзор сейчас, а не

где-то в далёком будущем и сейчас я в процессе тестирования столкнулся с рядом проблем. Начнём, хотя бы, с того, что значительная часть ПО, которое я использую для теста ноутбуков на Windows не работает на macOS в принципе. Поэтому мне пришлось скорректировать методику тестирования специально под новый Macbook. Благо, пока я её корректировал, я успел также кроссплатформенные тесты провести и на ноутбуках, которые я тестировал параллельно с Macbook, поэтому будет что и с чем сравнить.

Также у меня возникли проблемы при тестировании старым софтом для Mac. Этот момент я проясню подробно позже ниже в статье.

Предвещаю, что будет много вопросов «а почему ты это не протестировал или то не протестировал?». Господа, я 14 февраля написал в своём Telegram-канале, что мне приехал Macbook Pro с чипом M1 на тест и спросил, что бы Вы хотели, чтобы я протестировал. Обычно я не пишу, какие ноутбуки мне приезжают на тест, но о том, что у Стаса на канале выйдет обзор Macbook Pro с M1 Вы и так знали, да и на стриме с Чимой Стас позднее в тот же день чуть позднее сообщил, что я уже начал тестировать новый Macbook. И далее то, что я занимаюсь обзором на Macbook Pro с Apple M1 Стас озвучивал едва не на в каждом 2-ом своём стриме. После того, как я собрал первичный материал, я сел писать эту статью 5-го марта и за всё это время всего 7 человек мне написали либо вопросы личного характера (не для статьи), либо просили определённые тесты отразить в статье, и проконсультировали в проведении этих тестов. Остальные тесты я проводил из своего сугубо личного интереса. Я уже закончил писать этот текст и сдал Macbook Pro на M1 обратно Цифрусу, но Вы можете следить за моими публикациями в Telegram и, в следующий раз, когда я спрошу, что бы Вы хотели видеть в обзоре, например, если ко мне приедет Macbook на Apple M1X или Apple M2, пишите подробно, как мне проводить нужный Вам тест, почему этот тест важен и чем имеющиеся тесты не отражают той картины, которую призван отразить Ваш тест, и я обязательно учту Ваш запрос. Также можете свои пожелания по тестам ноутбуков с ОС Windows оставлять в комментариях к статьям на сайте или к записям в моём Telegram-канале.

Немного спойлерну Вам, что будет далее: этот обзор будет не только обзором как таковым, но и отчасти ответом на вопросы других блогеров, которые выпускали обзор на Mac с Apple M1, и которые сталкивались с проблемами в процессе его тестирования, либо же которые ошибочно утверждали о наличии в Apple M1 неких дополнительных технологий, подтверждения которым не нашлось. Так или иначе, нам будет что обсудить.

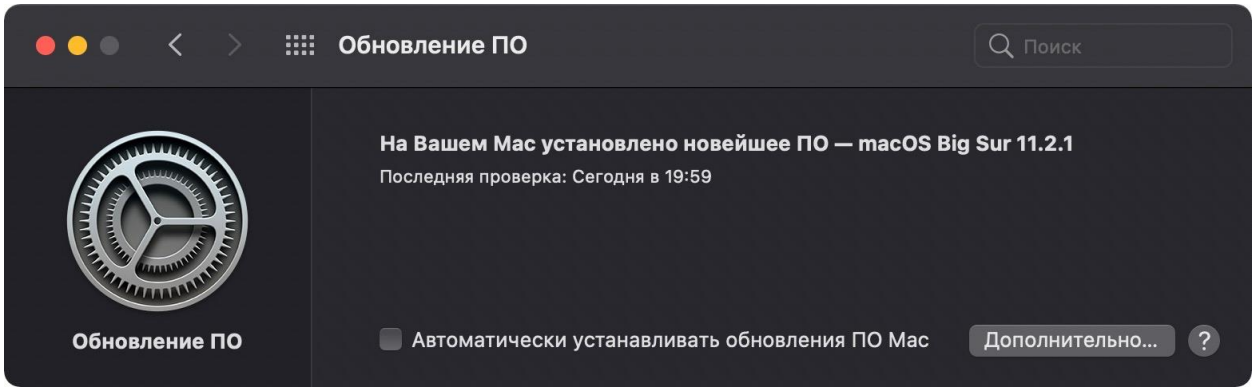
Выражаю огромную благодарность Кириллу Варшамову (kavars) за разработанное им приложение Benchmark для теста SSD и за помощь с тестами скорости компиляции проектов на Swift и Obj-C в XCode, Владимиру Барусу за содействие в тестах XCode и тесте Unigine Valley, а также Adatan`у за помощь с тестами скорости компиляции проекта Veloren на Rust.

Скажу прямо: я не питаю надежд, что эту статью хоть один человек прочитает полностью. Она вышла слишком массивной. В одной её части я защищаю Apple, т.к., на мой взгляд, её продукцию необоснованно раскритиковали в некоторых аспектах, а в другой я сам уже по делу раскритикую новый Macbook Pro, т.к. критиковать его действительно есть за что. При чтении одной части статьи Вам может показаться, что мне за этот обзор заплатили Apple, а при чтении другой – что заплатили конкуренты Apple. Жаль, но нет, ни одна компания мне за это не платила. Я буду как восхищаться гениальными решениями инженеров Apple в одних аспектах ноутбука, так и негодовать с ужаснейших решений в других аспектах.

С 5-го марта я печатал этот текст и параллельно продолжил проводить свои тесты вплоть до 10-го апреля, когда я отформатировал Macbook Pro на M1, переустановил на нём ОС и оставил его у Стаса для сдачи Цифрусу. Материала очень много за практически 2 месяца тестов. Надеюсь, что Вам будет интересно это читать. Не всё, понятное дело, я не рассчитываю, что Вы всё прочтёте, так что пользуйтесь оглавлением/закладками и знакомьтесь именно с теми аспектами, которые лично Вас интересуют. Какой бы именно вопрос Вас не интересовал, рекомендую Вам ознакомиться с выводами в конце, которые подытожат основные тезисы моего труда. На этом всё.

Потихоньку приступим к увлекательнейшей истории об Apple M1.

1. Перед началом тестов.



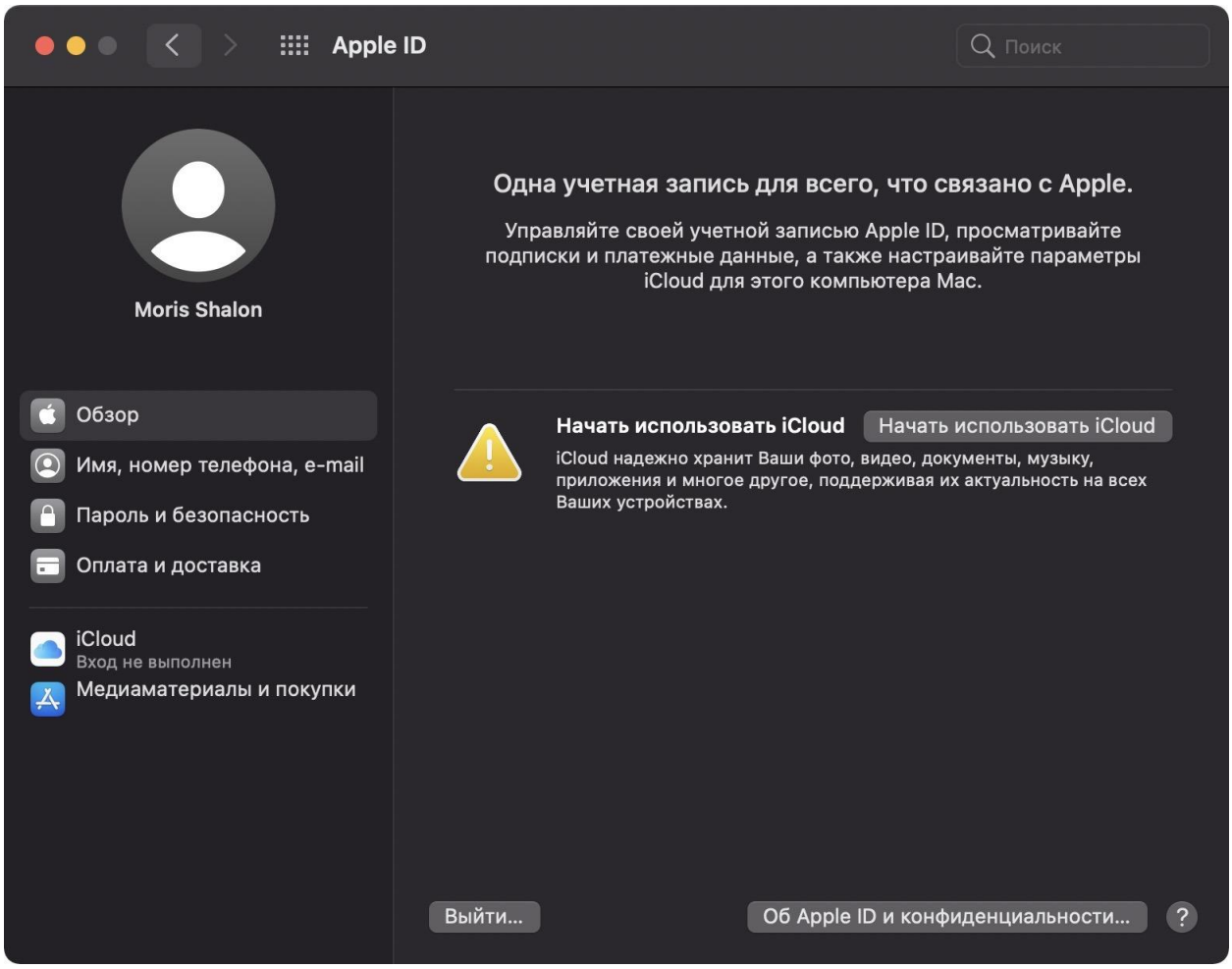
Скриншот 1. Обновление Macbook Pro (Apple M1) до macOS 11.2.1.

Для начала я обновил Macbook Pro до последней доступной для него на момент начала тестов версии macOS 11 Big Sur – 11.2.1.



Скриншот 2. Macbook Pro (M1). Об этом Mac.

По понятным причинам, серийный номер я стёр на скриншоте.



Скриншот 3. Macbook Pro (Apple M1). iCloud – вход не выполнен.

Также я не стал входить в iCloud. Почему? Да потому, что для меня крайне нежелательно, чтобы в ходе тестирования на результатах как-либо сказывалась фоновая активность iCloud. Авторизовался только в AppStore, т.к. только он мне он требовался для загрузки приложений. Тут также по понятным причинам, на скриншоте свой почтовый адрес потёр.



Скриншот 4. Xiaomi Mi Notebook Pro GTX. Об этом Mac.

Я также установил чистую macOS 11.2 Big Sur на заводской SSD от SK Hynix в моём основном рабочем ноутбуке Xiaomi Mi Notebook Pro GTX с Core i7 8550U (в модели GTX у него TDP 25 Ватт). Windows и Ubuntu остались нетронутыми на моём SSD Samsung 970 Evo Plus.

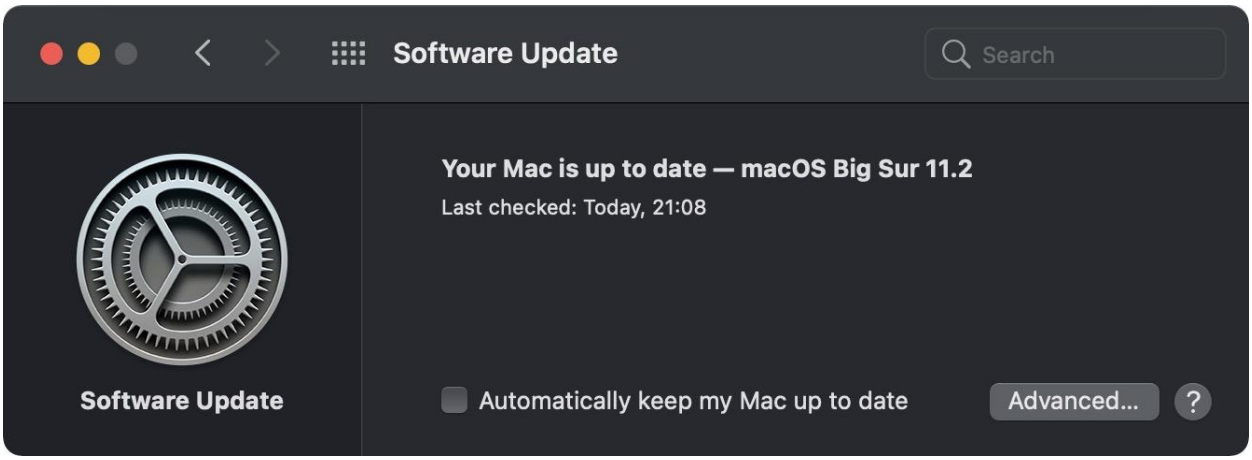
Чтобы исключить влияние стороннего софта на результаты тестов, все тесты я провёл на чистой ОС, на момент тестирования у меня были установлены только требуемые для данного теста приложений (и Telegram).

Зачем я вообще накатил себе хакинтош? Почему мы сравниваем новый Macbook Pro на базе Apple M1 именно с Xiaomi? Причин 2:

1. Он у меня под рукой. У меня своих ноутбуков всего 2: Asus Q500A (2013), и Xiaomi Mi Notebook Pro GTX (2018) (i7 8550U и GTX 1050 Max-Q).

2. Он по производительности процессора (без учёта графики) не отличается от ноутбука моего хорошего друга, Кирилла, Macbook Pro 13 2018 Core i5 8259U и 4 портами Thunderbolt 3, в чём вы убедитесь далее в статье. Считайте, что таким образом я сравниваю CPU в Macbook Pro на Intel и M1.

Уточню: я никого не призываю устанавливать хакинтош. Я установил его исключительно ради сравнения производительности с новым Macbook Pro. Если желаете работать на macOS, купите себе компьютер от Apple.



Скриншот 5. Xiaomi Mi Notebook Pro GTX. Установлены все доступные обновления.

Почему установил 11.2, а не более свежую? Обновления 11.2.1, 11.2.2 и 11.2.3 (вышли, когда я уже сел писать статью после всех тестов) не доступны для моего хакинтоша. Впрочем, разница между 11.2 и 11.2.3 не критична.

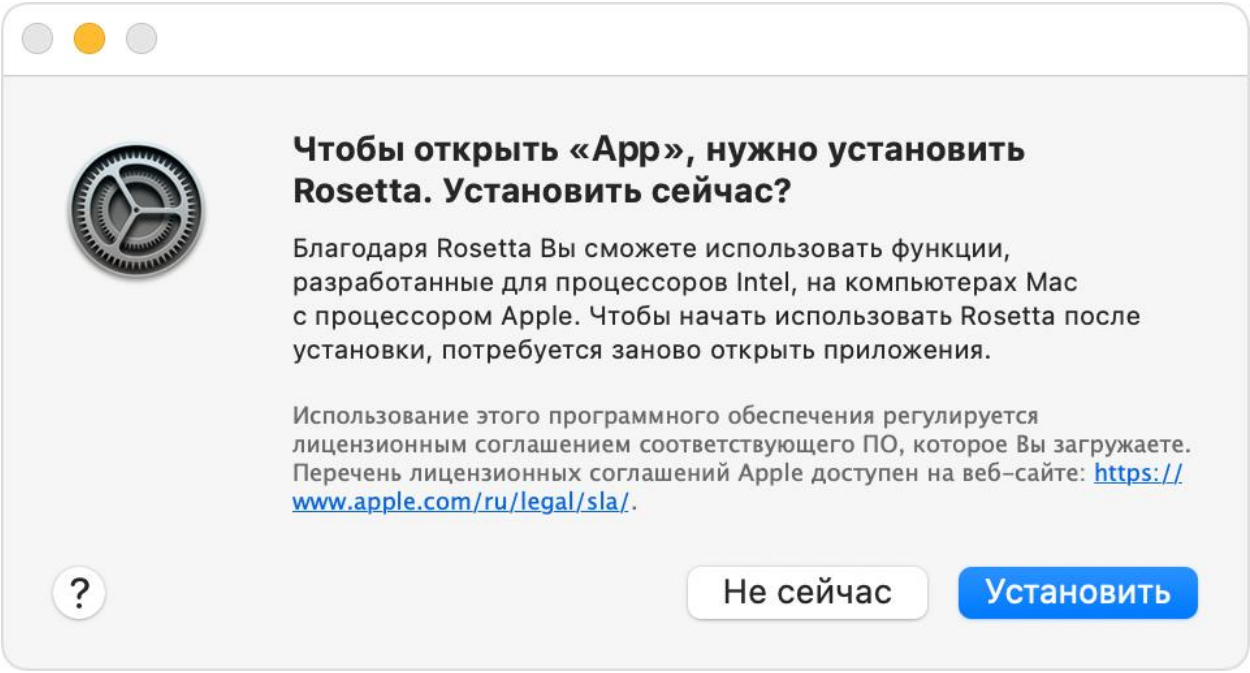
Почему же я говорю «без учёта графики»? У Apple до недавнего времени был заключен эксклюзивный контракт с Intel, в рамках которого Intel поставляли Apple первыми свои SoC с индексом U с повышенным TDP (28 Ватт вместо стандартных для U чипов 15 Ватт) и более производительными интегрированными графическими ускорителями. В моделях с 4 портами Thunderbolt 3 устанавливались модели чипов Intel с расчётным TDP 28 Ватт (например, i5 8259U, с которым сопоставим мой 8550U в Xiaomi, или i5 1038NG7, с которым также будет сравнение в нескольких тестах), тогда как в модели с двумя портами Thunderbolt 3 (в случае с 2019 и 2020 годами) или более старыми моделями без TouchBar (в случае с моделями 2016 и 2017 годов) устанавливались процессоры с расчётным TDP 15 Ватт (например, 8267U или 7660U). Впрочем, что у моделей с расчётным TDP 28 Ватт, что у моделей с расчётным TDP 15 Ватт, производительность интегрированной графики была существенно выше, чем у аналогичных чипов от Intel для ноутбуков на Windows, раза в 1.5-2 примерно. Моя видеокарта 1050 Max-Q не функционирует на macOS из-за отсутствия поддержки технологии NVidia Optimus операционной системой от Apple, потому по графике Xiaomi будет слабее, чем Macbook Pro 13” 2018 с 4x TB3. Однако на тестах CPU части в SoC Intel это никак не скажется.

Ну, что с чем сравниваем разобрались, приступим потихоньку.

2. Rosetta 2 и чем отличаются macOS для ARM и x86_64.

2.1. В чём суть Rosetta 2 и чем она отличается от эмулятора.

Первое, что меня заинтересовало в новом MacBook Pro с чипом Apple M1, это то, какие же изменения были внесены Apple для полноценной работы их Операционной Системы macOS на своём собственном чипе Apple M1.



Скриншот 6. Изображение с сайта Apple. Сам не заскринил этого оповещения. Оно появляется лишь раз.

Я думаю, Вы и так слышали в других обзорах про то, что при запуске x86_64 приложений на macOS, Вам система предлагает сразу же установить Rosetta 2 для беспроблемной (спойлер – нет) работы с x86_64 приложениями.

Но как оно вообще работает? И почему трансляция кода, выполняемая Rosetta 2 при первом запуске (или до первого запуска) приложения x86_64 на компьютере с архитектурой ARM, это не то же самое эмуляция платформ x86 и x86_64, последнюю из которых недавно внедрили в Insider Preview версию Операционной Системы Windows 10 on ARM от Microsoft? В чём отличие?

Начнём с того, как работает эмулятор. Эмулятор — это программное обеспечение, которое, очень упрощённо говоря, создаёт программную модель процессора (или ещё и Операционной Системы), которая работает внутри Вашей действующей ОС, запущенной на Вашем действующем процессоре. На то, чтобы создать рабочую имитацию нужной архитектуры и заставить в ней работать весь софт, тратится огромная часть ресурсов ЦП.

Транслятор кода не создаёт виртуальную модель Вашего процессора. Вместо этого он транслирует уже готовый машинный код под одну архитектуру в готовый машинный код для другой архитектуры. Возможная грубая аналогию: это как перевести текст с немецкого на итальянский.

Более простая для понимания, но очень грубая аналогия: вместо того, чтобы, шить игрушки для игры с моделью игрушечной радиоуправляемой машины, подстраиваясь под габариты этого автомобиля, Вы воссоздаёте эту машину в нормальных для человека размерах и сами садитесь за руль. Вот в чём отличие транслятора от эмулятора. Радиоуправляемый игрушечный автомобиль никогда по скорости не обгонит аналогичную модель нормального автомобиля. Пожалуйста, умелые специалисты, простите мне эти чрезмерно грубые аналогии. Я не имел и не имею ни малейшего желания задеть этим чьи-либо чувства. Я хочу лишь сделать статью как можно более подробной и, при этом, как можно более понятной большинству читателей.

Что особенно важно: у Apple мало того, что используется трансляция машинного кода с одной архитектуры под другую, которая сама по себе эффективнее эмуляции конечной платформы, так эта самая трансляция ещё и применяется предварительно, т.е. приложение у Вас заранее при первом запуске (или до первого запуска) программы x86_64 переводит её машинный код под x86_64 в аналогичный машинный код под ARM64 (aarch64).

С одной стороны, это не хорошо, ведь у Вас на накопителе выделяется место для хранения как исходной версии программы под x86_64, так и транслированной версии для нативного запуска на aarch64 (ARM64) CPU. С другой стороны, заранее транслировать программу это в разы более эффективно, чем «на лету» переводить каждый x86 вызов программы в нужный машинный код для ARM, и тем более эффективнее эмуляции. Так, потеря производительности при использовании Rosetta 2 составила всего 20-30%, хотя есть и исключения с просадкой в 70%+. Также, если в Rosetta 2 выявят ошибку, Apple смогут обновлением исправить её и очистить у Вас все кэши Rosetta 2. Тогда при следующем запуске Ваше ПО будет транслировано заново и ошибки в нём, связанные с ошибкой в Rosetta 2, будут исправлены.

Да, у Вас долго устанавливаются приложения или долго запускаются в первый раз после их установки или обновления, но это только в первый раз. Во второй раз приложения запускаются, как минимум, не медленнее чем на аналогичном по цене прошлом поколении Macbook Pro с SoC Intel, да и задачи в самих приложениях выполняются значительно быстрее.

Если Вы смотрели видео Linus Tech Tips с обзором Mac Mini, то на 14:45 слышали, что Apple M1 умеет запускать ещё и приложения для x86 при помощи встроенного в процессор транслятора кода. Но Apple не делали там ничего такого, Rosetta 2 – чисто программное решение. Нет подтверждения твиту, на который Linus ссылается в видео. Просто фейк дошёл и до Linus`a.

2.2. Ограничения Rosetta 2. AVX инструкции.

Какие же ограничения есть у Rosetta 2? Их несколько:

1. Не поддерживается трансляция кода, эксклюзивного для процессора Intel. Например, не транслируется код, задействующий технологию виртуализации Intel VT-x. У чипов Apple Silicon есть своё решение для виртуализации. Также не транслируются, к примеру, запросы температурных датчиков процессоров Intel. У Apple Silicon есть альтернативное решение.
2. Хоть и транслируется код x86_64, но не весь, нет поддержки инструкций AVX, AVX2 и AVX512, которые задействуются даже в играх ещё с 2013 года. Хотя, в свежем Cyberpunk 2077, к примеру, имеется лишь проверка на наличие поддержки AVX инструкций, а сами они не задействуются, так что их проверку спокойно можно отключить, пропатчив исполняемый файл игры.
3. Не транслируются расширения ядра. Если Вы в работе используете старое ПО, которое требует для работы установки расширения ядра (kernel extension или kext, аналог драйвера в Windows и Linux), которые ещё не были переписаны под ARM, то ничего работать не будет. Например, старые версии Lulu (файрволл с открытым исходным кодом) работать не будут, а вот с новыми версиями у Вас проблем возникнуть не должно.

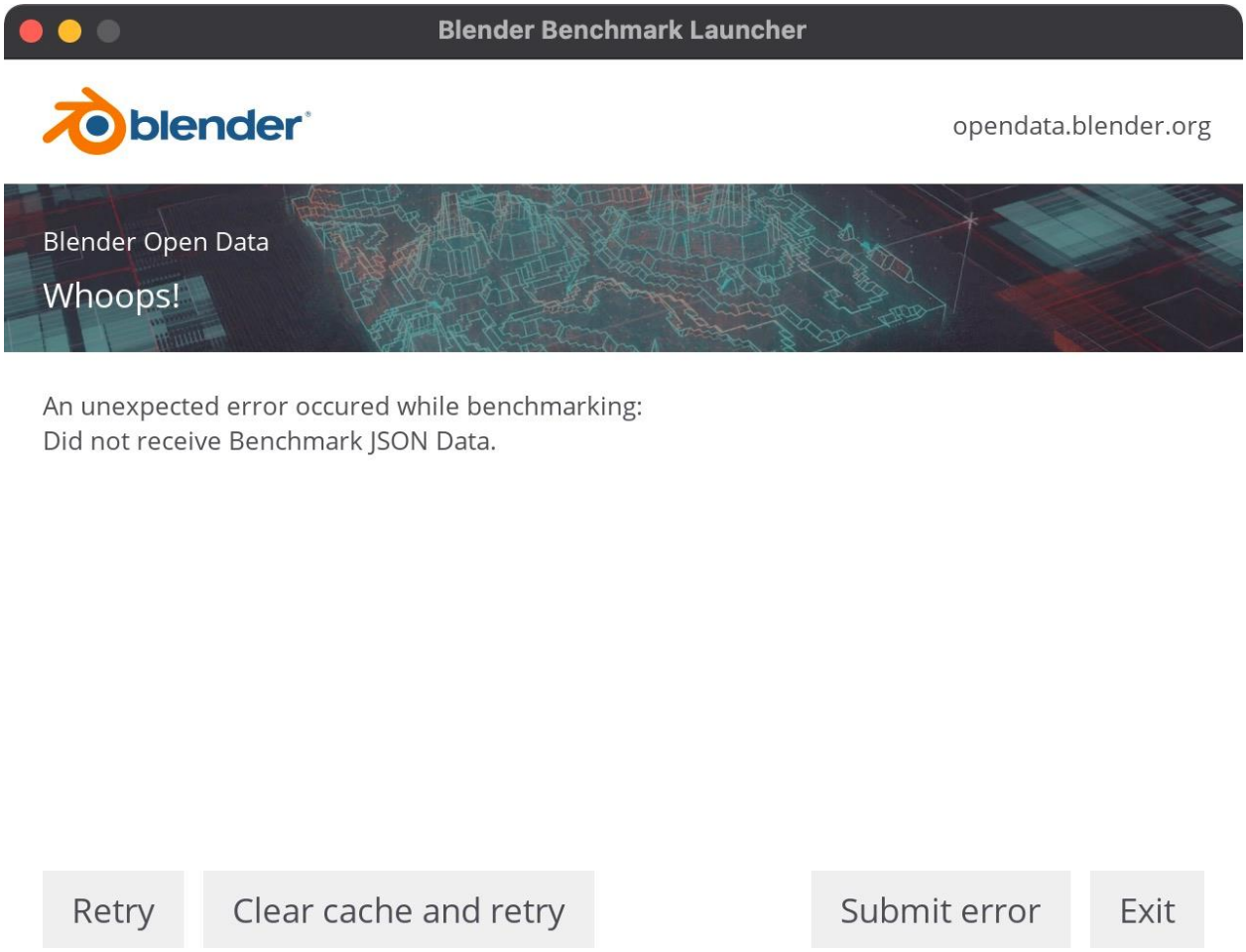
О том, что Rosetta 2 имеет вышеописанные ограничения, Apple сами пишут на своём официальном сайте.

Проблема в том, что многие люди не прочитали этого на оф. сайте Apple и стали винить Apple во лжи о беспроблемной работе Rosetta 2.

Приведу пример с видео обзором Macbook Pro и Macbook Air на M1 от Романа Ястреба, т.к. у него, на мой взгляд, один из самых подробных разборов Macbook Air и Pro с чипом M1 в российском сегменте YouTube. Потому и примеров с неточностями в его видео, которые присущи и другим блогерам, будет много.

Уточню: я со всем уважением отношусь к Роману, и я никак не пытаюсь его обидеть или оскорбить. Я с удовольствием смотрю видео на его YouTube-канале и делюсь ими с друзьями, когда в его видео присутствует информация, важная для них. Этот текст – всего лишь ответы на его вопросы.

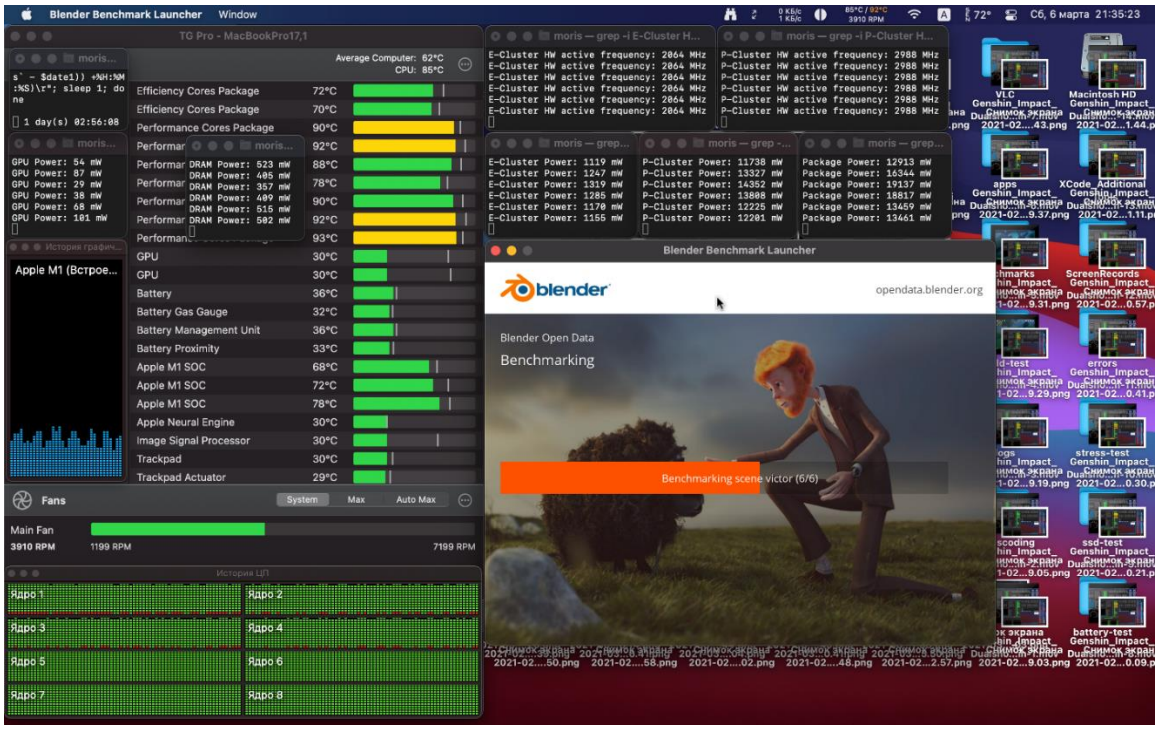
На 36:35 у Романа проблемы с Blender 2.90, а с 2.83 у него всё хорошо.



Скриншот 7. Macbook Pro 13 2020 (Apple M1). Blender Benchmark 2.04 (Blender 2.90). Ошибка при запуске теста.

Я проверил, и да, действительно с версией 2.90 тест провести не удаётся. Однако этому есть объяснение. Первое, что пришло мне на ум, что Blender мог в версии 2.90 использовать на Macbook с M1 AVX инструкции, которая Rosetta 2 транслировать не способна. Я нашёл информацию на форуме Blender: движок для рендеринга Cycles использует AVX инструкции. Помимо этого, о проблемах с версией 2.90 на Intel сообщали тут, тут и тут, и точно также у людей версия 2.83 работала, проблемы были именно с 2.90.

После я выяснил, что одна из библиотек (составных компонентов исходного кода Blender), OIDN, ранее неверно проверяла наличие поддержки инструкций AVX-512 процессором, и в итоге программа пыталась задействовать эти инструкции даже на процессорах, которые не имели их поддержки, что могло приводить к падению программы на macOS и на старых Intel. Однако позднее проблему решили обновлением OIDN.



Скриншот 8. Macbook Pro 13 2020 (Apple M1). Blender Benchmark 2.04 (Blender 2.91.2). Рендеринг сцены victor.

Как Вы можете видеть, с новой версией Blender 2.91.2, которая вышла уже после исправления ошибок в её библиотеках, подобных проблем уже не возникло. Тем не менее, может возникнуть следующий вопрос: если с AVX инструкциями столько проблем, то зачем их используют?

Дело в том, что использование AVX инструкций может значительно повысить производительность компьютера в определённых задачах.

Версия	Реализация	Время выполнения (с)	GFLOPS	Абсолютное ускорение	Относительное ускорение	Процент производ
1	Python	25552,48	0,005	1	—	0,00
2	Java	2372,68	0,058	11	10,8	0,01
3	C	542,67	0,253	47	4,4	0,03
4	Параллельные циклы	69,80	1,97	366	7,8	0,24
5	Парадигма «Разделяй и властвуй»	3,80	36,18	6727	18,4	4,33
6	+ векторизация	1,10	124,91	23224	3,5	14,96
7	+ интристики AVX	0,41	337,81	52806	2,7	40,45

Скриншот 9. Публикация компании VDSina на habr.com. Разница во времени выполнения операции с оптимизацией кода.

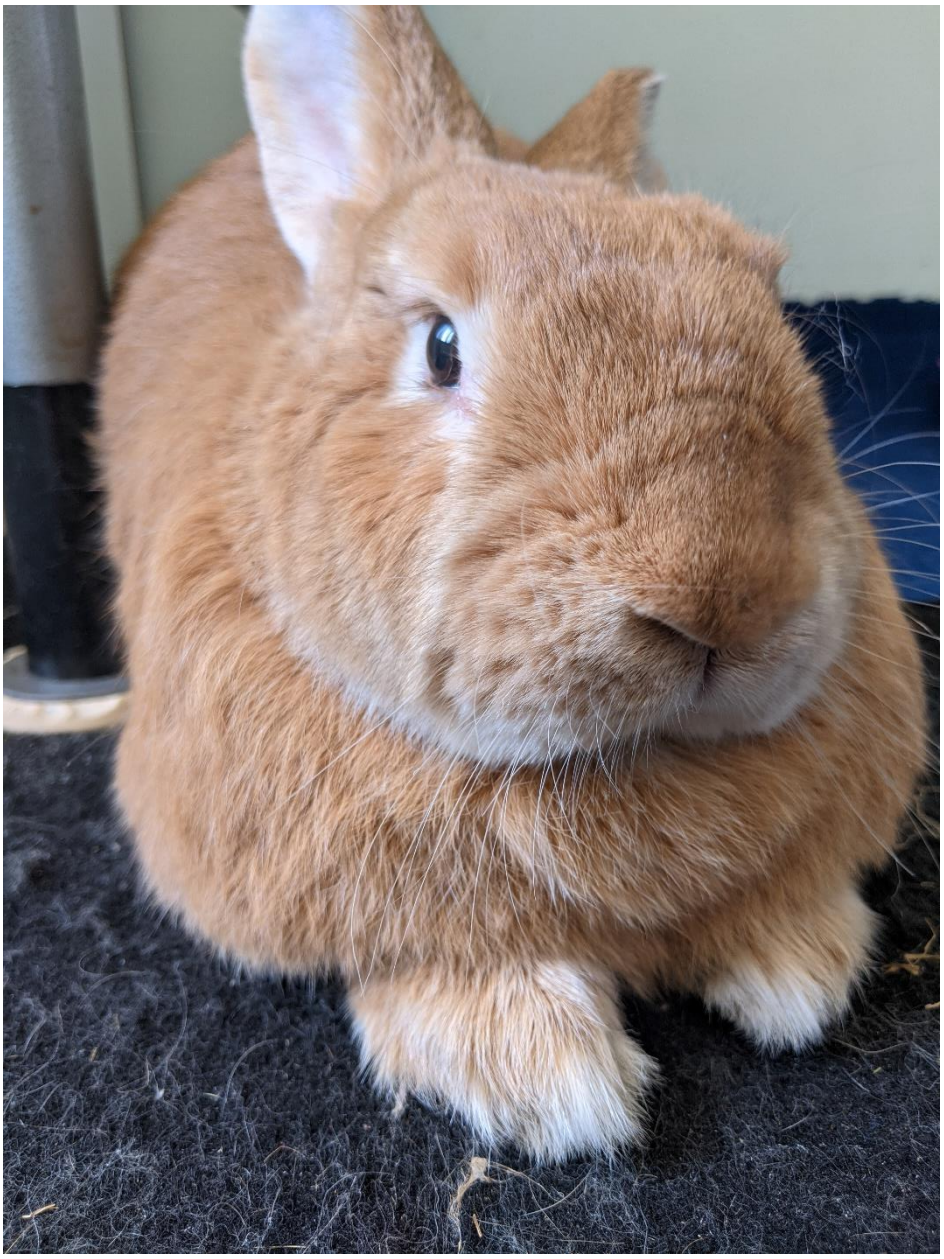
Выше приведён скриншот из публикации компании VDSina на habr.com «Как перезапустить закон Мура программными методами. Ускорение софта в тысячи раз», раздел «Методы ускорения программного обеспечения». Рекомендую ознакомиться с данной статьёй. Как Вы можете видеть, в задаче перемножения двух матриц 4096x4096, распараллеливание задачи на все ядра/потоки (версия 4), грамотное применение иерархии кэшей процессора (версия 5), векторизация вычислений (версия 6) и применение AVX инструкций (версия 7) позволили сократить время, затраченное на выполнение задачи по перемножению матриц, с 542,67 секунд на языке C без этих оптимизаций до 0,41 секунды со всеми оптимизациями. Рост в скорости выполнения операции составил 1323,58 раза или же, иначе выражаясь, прирост составил 132258%. Одни только AVX инструкции повысили скорость выполнения поставленной задачи в 2,68 раза.

С Python я не сравниваю, т.к., несмотря на то, что код на Python можно скомпилировать, всё равно, прежде всего, это интерпретируемый язык и очевидно, что по скорости он не сможет сравниться со скомпилированным C.

Именно с целью грамотной оптимизации и задействования всех ресурсов процессоров с архитектурой x86, и используются AVX инструкции.

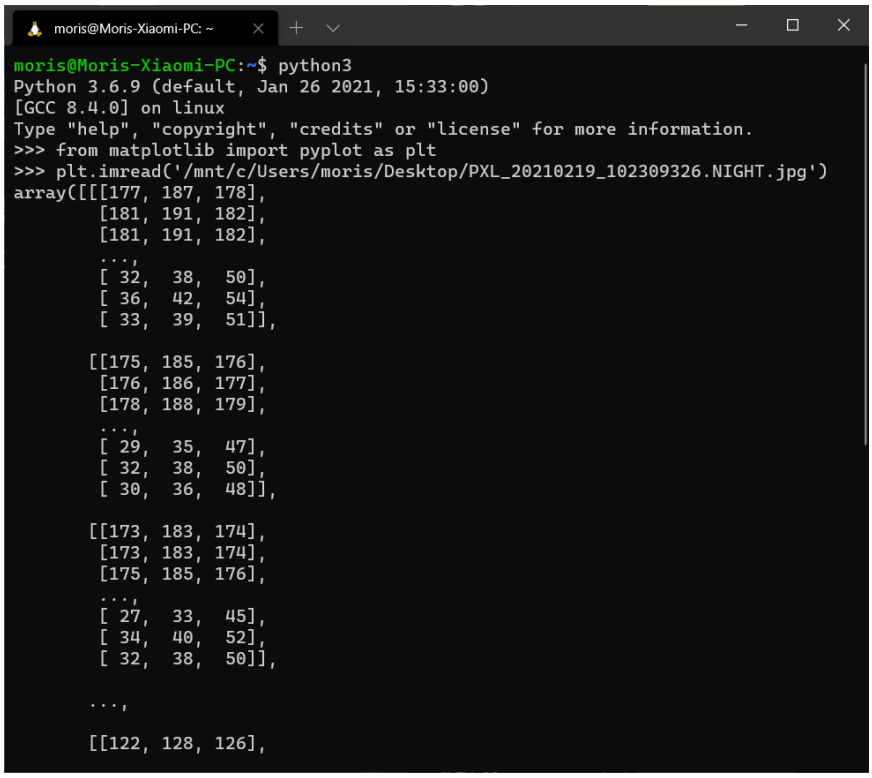
У ARM64 процессоров также имеется набор специфичных инструкций (SIMD - single instruction, multiple data) NEON и SVE (Scalable Vector Extension), грамотно используя которые можно добиться повышения скорости обработки чисел с плавающей запятой (например, FP16, примером такого числа будет 1,0009765625). Эти инструкции также можно задействовать для повышения производительности системы, к примеру, в задачах по перемножению матриц.

Что я тут пишу про задачи с матрицами? Кому какое до них дело? Мы же в реальности с ними не сталкиваемся, верно? Вообще-то нет, Вы с этим примером каждый день сталкиваетесь в быту. Изображения - ведь тоже матрицы. У вас X пикселей по горизонтали и Y пикселей по вертикали, и обработка изображений, а если изображение ещё и цветное, то каждый пиксель хранит несколько значений (R насыщенность красного цвета, G насыщенность зелёного цвета и B насыщенность синего цвета, а может быть и прозрачность A, альфа-канал, а может быть другое цветовое пространство и т.д.). Значит, изображение – это трёхмерный массив (матрица), состоящий из X пикселей по горизонтали, Y пикселей по вертикали и Z данных каждой ячейки в этой двухмерной плоскости (цвета r, g, b и прозрачность a).



Скриншот 10. Фотография моего кролика, которую можно представить в виде трёхмерного массива в matplotlib.

Выше я привожу фотографию моего кролика, которую можно точно также представить в виде огромного трёхмерного массива, состоящего из 3024 пикселей по горизонтали, 4032 пикселей по вертикали, и каждый пиксель при этом содержит информацию о 3-ёх цветах: красный (Red), зелёный (Green) и синий (Blue). Фотография сохранена в .jpeg, информация о прозрачности (альфа-канале) отсутствует, поэтому каждый пиксель хранит лишь 3 значения, а не 4.



Скриншот 11. Python в WSL2. Чтение данных в изображении кролика как массива.

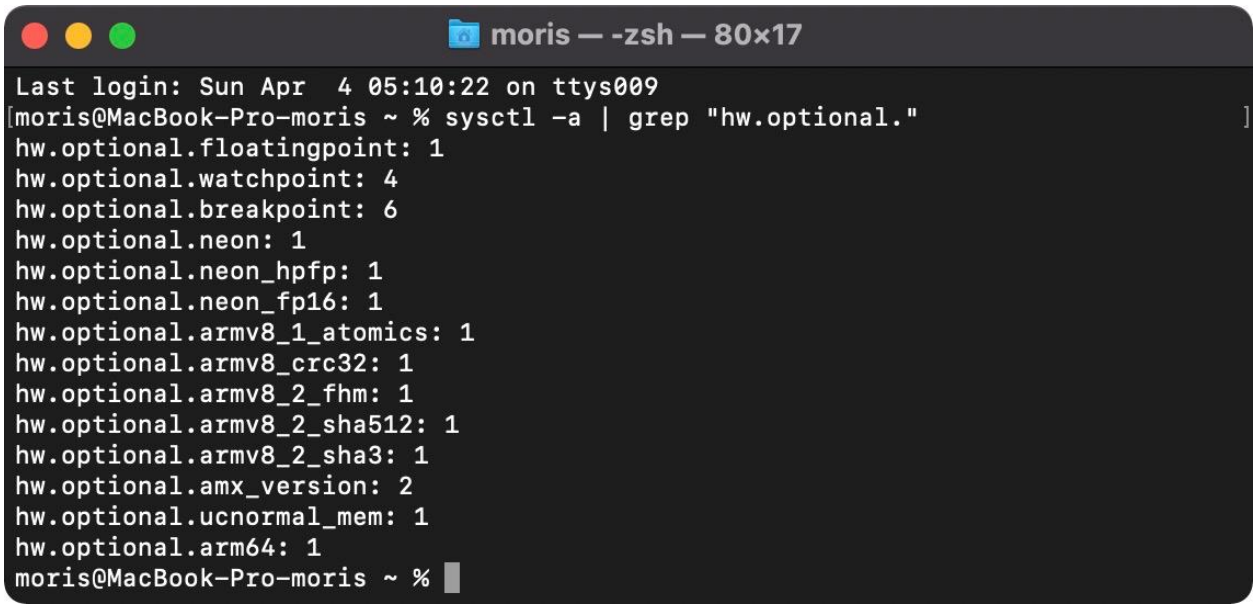
В языке программирования Python, к примеру, используя pyplot, модуль библиотеки matplotlib для работы с графикой и изображениями, можно прочесть изображение как массив, в котором каждый набор из 3 чисел внутри одинарных квадратных скобок – это 1 пиксель, а каждый набор из наборов таких чисел, скрытый за двойными квадратными скобками – строка, и так строки идут одна за другой 4032 раза, сколько и пикселей по вертикали.

Поскольку изображения можно представить как матрицы, то и AVX-инструкции могут значительно ускорить процесс их обработки, к примеру, при наложении эффекта размытия по Гауссу. Изображения – лишь одни из множества примеров, AVX-инструкции много где можно задействовать.

Разработчики Blender стараются задействовать все аппаратные возможности Вашей системы для того, чтобы ускорить Ваши вычисления. А с версии 2.91.2 была исправлена проверка на наличие AVX инструкций. Если они не поддерживаются, они и не будут задействованы, как и положено.

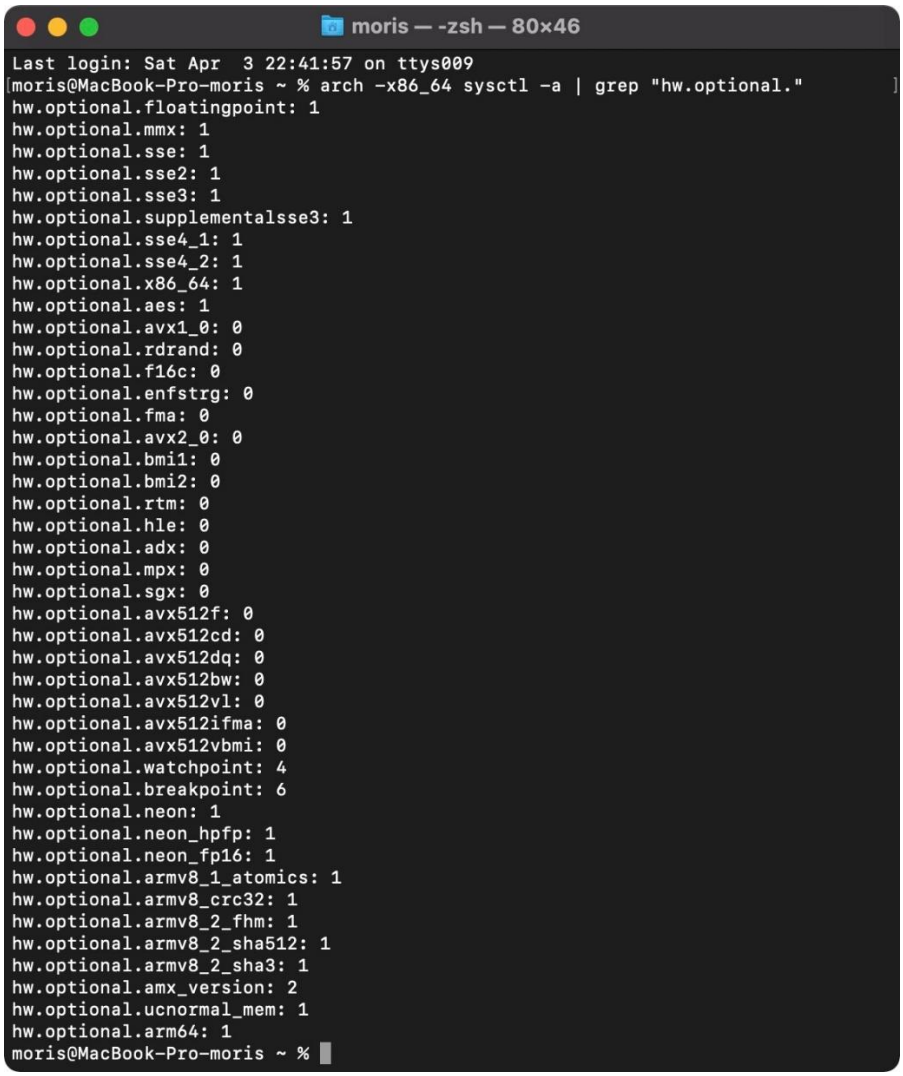
Как проверить, какие расширения наборов инструкций поддерживает Ваш процессор на компьютере под управлением macOS? Apple также дали ответ на этот вопрос. Надо просто в Terminal ввести следующую команду:

```
sysctl -a | grep "hw.optional."
```



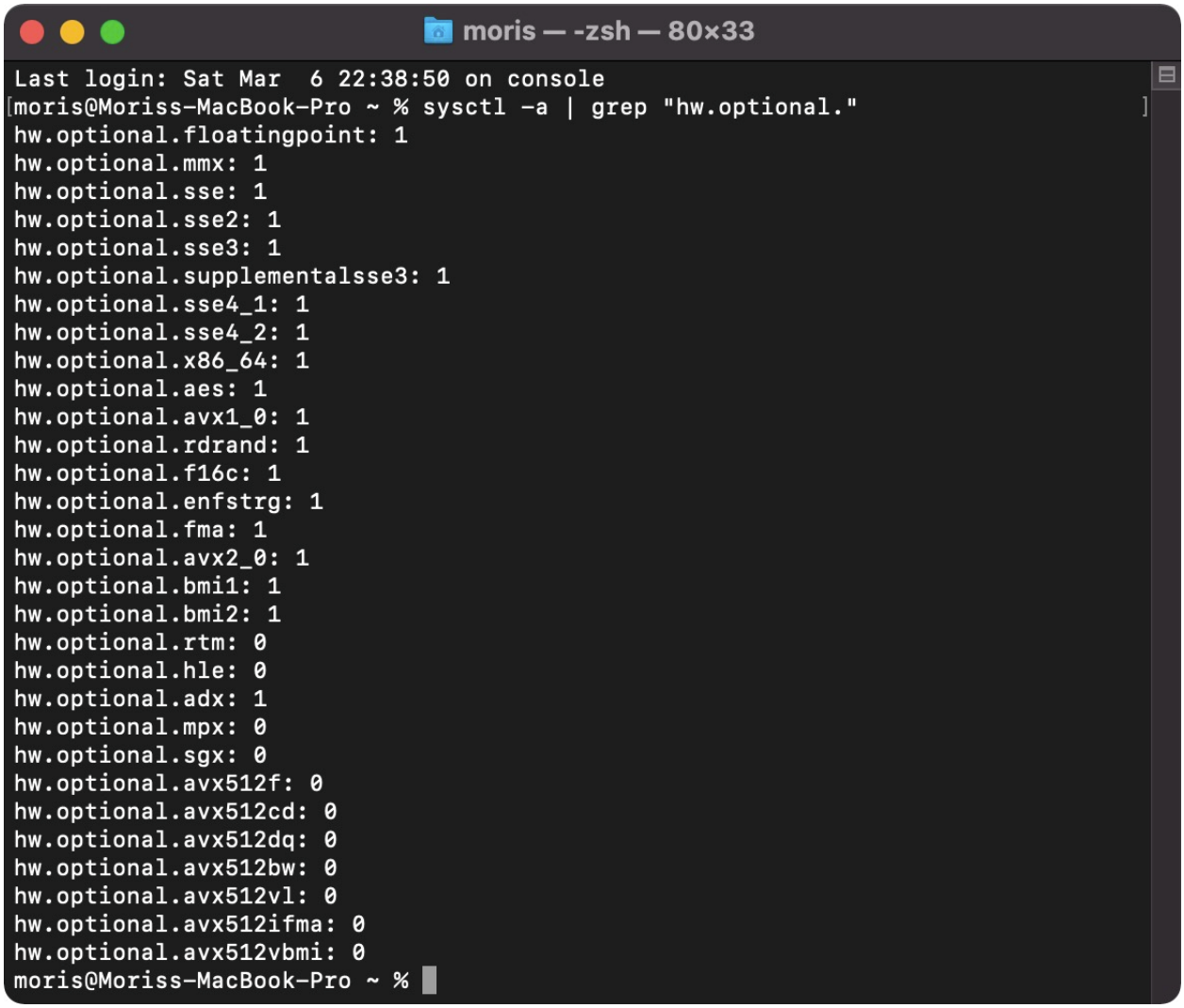
Скриншот 12. Macbook Pro 13 2020 (Apple M1). Доступные для использования в ПО расширения инструкций ARM.

Как видите, на Macbook Pro с чипом M1 можно использовать инструкции NEON для операций с плавающей запятой (float, fp16), но нельзя использовать набор расширений инструкций AVX для x86_64 платформы.



Скриншот 13. Macbook Pro 13 2020 (Apple M1). Доступные для использования в ПО расширения инструкций с Rosetta 2.

Если выполнить ту же команду через Rosetta 2, то мы увидим в числе доступных x86 расширений инструкций и MMX, и SSE, но не AVX. Занятно, при использовании Rosetta 2 нам всё ещё доступны инструкции ARM64, так что разработчик может их принудительно задействовать даже через Rosetta 2.



Скриншот 14. Xiaomi Mi Notebook Pro GTX. Доступные расширения инструкций.

Можете сравнить информацию на выходе с таковой с моего Xiaomi Mi Notebook Pro GTX. Где после двоеточия 0, значит, что поддержка указанного набора инструкций отсутствует, а где 1 – присутствует.

Как Вы видите, в случае с SoC i7 8550U, который установлен в моём ноутбуке Xiaomi, присутствует поддержка расширений инструкций AVX и AVX2, но отсутствует поддержка AVX-512. Это значит, что программы, которые имеют в своём составе код, задействующий инструкции AVX-512, точно также не будут работать у меня на моём Xiaomi, как и на Macbook Pro с чипом M1, точно также как и на Macbook Pro 13 2018 с SoC i5 8259U.

Поддержка инструкций AVX впервые в серии SoC Intel Core дебютировала в 2011-ом году с релизом 2-го поколения (Sandy Bridge), а поддержка AVX2 впервые была внедрена в 4-ом поколении (Haswell, 2013).

AVX-512 же впервые в ноутбуках с Intel Core дебютировали в 10-ом поколении Ice Lake, которое было произведено с применением 10 нм техпроцесса. В то же время 10-е поколение Comet Lake, которое было произведено по 14 нм нормам техпроцесса, лишено поддержки AVX-512.

У Apple на базе 10-го поколения Ice Lake (10 нм) было выпущено всего 2 компьютера: это ноутбуки Macbook Air 2020 и Macbook Pro 2020 с 4 портами Thunderbolt 3. Очевидно, что, т.к. Apple теперь отказываются от Intel, да и прошло довольно немного времени с релиза этих ноутбуков, софта под macOS, задействующего AVX-512, было выпущено крайне мало. Да и навряд ли такого софта станет больше в будущем, раз уж Apple отказывается от AVX с уходом на ARM. Вот только поддержка AVX2 присутствует в компьютерах Apple с середины 2013-го года, а AVX (AVX1) так и вовсе с середины 2011-го года. За 8 или 10 лет, очевидно, что ПО, задействующего AVX инструкции, могло набраться довольно немало под macOS. Или нет?

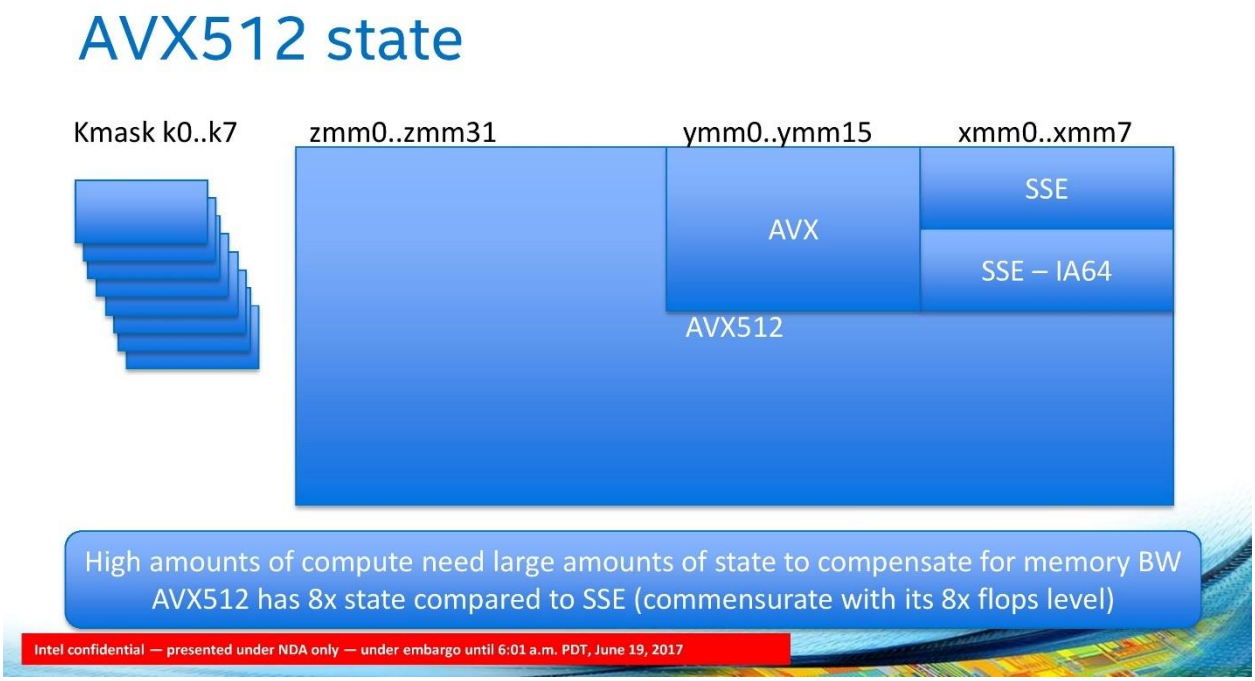
Сильно по этому поводу переживать не стоит, ведь для того, чтобы грамотно задействовать AVX инструкции, разработчику нужно разбираться в Ассемблере или, как минимум, нужно понимать, как работают векторы, как эффективно обрабатывать большие массивы данных, и желательно ещё понимать, как, написав такой софт, грамотно задействовать в нём оптимизации компилятора. Сложность задачи по написанию подобного ПО весьма высока, потому его меньше 1% среди общего числа программ. Да и, как правило, если разработчик прописывает сценарий для работы программы с AVX, он прописывает и сценарий на случай, если у компьютера нет поддержки AVX. Тот же Blender может работать и без AVX инструкций, проблема в версии 2.90 была лишь в неправильном определении наличия их поддержки, что было исправлено в версии 2.91.2, о чём я писал выше.

Но да, проблемы возможны, такое имеет место. Впрочем, как Вы видели, разработчики Blender эту проблему исправили, даже не адаптируя

ПО специально под Apple Silicon (ARM), а просто устранив неполадку в своём решении под x86. Теперь Blender нормально работает через Rosetta 2.

2.3. В чём сложность трансляции AVX-инструкций.

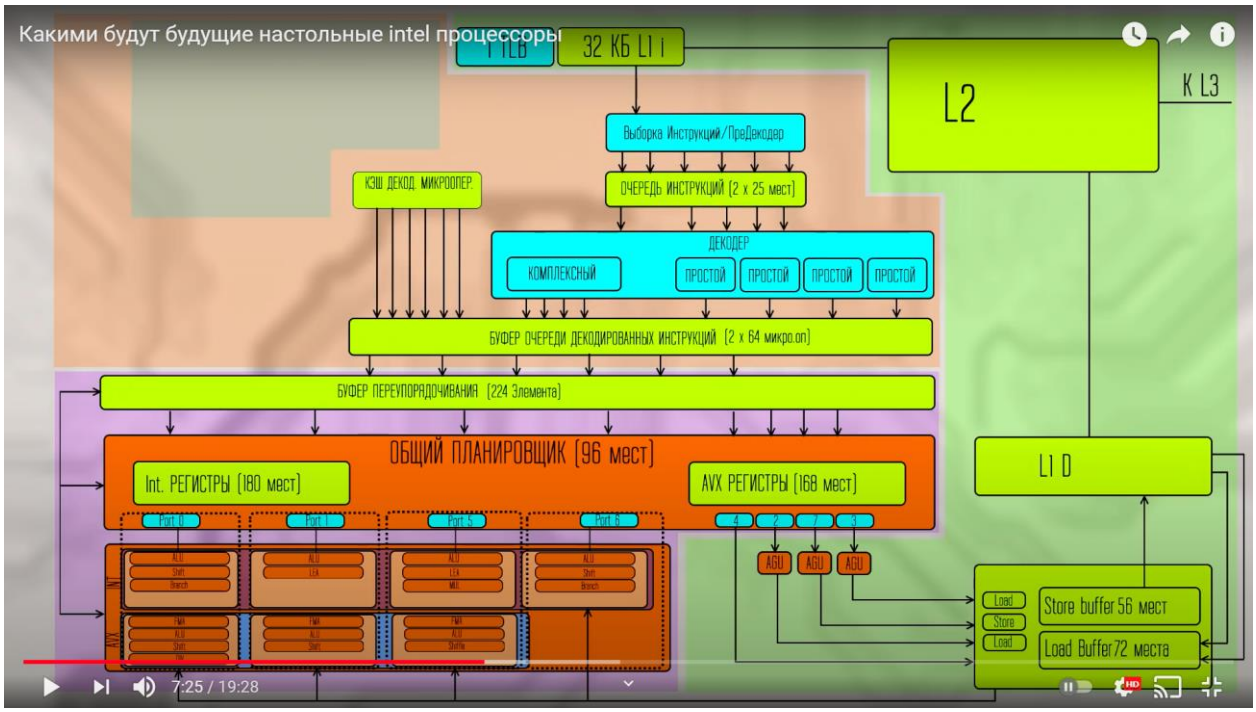
Для того, чтобы понять, почему Apple не реализовали трансляцию AVX инструкций, нужно узнать, что отличает AVX инструкции от старых SSE, и в чём суть внутреннего дизайна процессоров с AVX. Рассмотрим на примере CPU Intel. Это сложный материал, но постараюсь пояснить просто.



Скриншот 15. Разница в размерности данных, обрабатываемых за раз инструкциями SSE и AVX.

Главное, что отличает AVX-инструкции от других расширений инструкций, таких как SSE, это объём данных, за раз обрабатываемых инструкциями. Регистры, хранящие данные для AVX инструкций имеют размер в 256 бит, а в случае с AVX-512 – 512 бит. Благодаря этому AVX1 и AVX2 позволяют добиться существенно больших скоростей в вычислениях с плавающей запятой, а AVX512 позволяет повысить эффективность и целочисленных вычислений (64-bit Integer). Главная идея тут в том, чтобы максимально загрузить процессор, т.е. заставить его за каждую инструкцию обрабатывать больше однотипных данных параллельно. Это особенно эффективно в задачах рендеринга и шифрования (хотя, для AES есть ещё своё отдельное расширение набора инструкций).

Но за счёт чего удалось этого добиться и как вообще всё это дело работает? Тут есть 2 аспекта: новые исполнительные блоки и регистры. Ключевым моментом для нас в рамках статьи являются как-раз регистры.



Скриншот 16. Упрощённая схема структуры процессора от автора [YouTube-канала Этот Компьютер](#).

Дело в том, что инструкции выполняют операции с данными, которые где-то нужно хранить, и специально для большего числа однотипных (8x 32-бит для AVX-256) данных отведены 256-бит регистры под AVX-инструкции.

При исполнении AVX-инструкций, когда процессор нагружается по максимуму, стараясь за раз обработать как можно большее число данных, его тепловыделение значительно поднимается. Поэтому при наличии лимита по TDP, процессор при задействовании AVX инструкций, когда он нагружается по максимуму, как можно меньше простаивая, снижает свою тактовую частоту. И даже так он остаётся в разы производительнее благодаря AVX.

Регистры на 256 бит– это не единственные, но одни из главных отличий современных x86 процессоров от ARM. Не отличие RISC архитектуры с набором сокращённых по своей длине машинных команд от CISC с возможностью обработки широких команд, а именно AVX. Если Вы обратите внимание на схему выше из видео «Какими будут будущие настольные Intel процессоры» на YouTube-канале «Этот Компьютер», то заметите, что на ней также присутствует ещё и декодер, задача которого –

разбивать большие инструкции на микро-операции, которые уже понимают ядра процессора. Эти микро-операции являются RISC-подобными, и Intel ещё с Pentium Pro (1995 год) и AMD с K5 (1996 год) используют такой подход.

Однако дело ещё в том, что у x86 процессоров до 2011 года не было регистров на 256 бит и более, и именно поэтому у x86 процессоров для работы с регистрами и появились новые инструкции, которые позволяют задействовать эти самые регистры. Их то и назвали AVX инструкциями.

И главное, что мешает Apple реализовать поддержку этих инструкций – отсутствие регистров на 256 бит и более в архитектуре процессоров ARM. Дело в том, что по условиям ISA лицензии ARM, Apple не покупает готовый дизайн ядер, а сама их разрабатывает, задействуя в продукте набор команд ARM и базис архитектуры в целом. Apple не может вносить изменения в этот самый набор команд, поэтому даже внедрив регистры на 256 бит, они не смогут их задействовать. А максимальная ширина регистров у ARM – 128 бит, их можно эффективно задействовать при помощи SVE (Scalable Vector Extension), который я упоминал в главе 2.2, но полноценного аналога AVX-инструкций, в архитектуре ARM не имеется, т.к. и регистров на 256 бит нет.

Именно поэтому возникает сложность с трансляцией AVX кода, который на современных x86 CPU исполняется нативно безо всяких проблем.

В то же время, если сравнивать x86 и ARM, есть одно существенное преимущество у ARM: В ARM, как и в x86 процессорах, также используется декодер для разбивки команд на микро-операции, но у ARM для этой задачи задействуется уже не микрокод внутри процессора, а аппаратные блоки управления, которые значительно быстрее справляются со своей задачей. Т.е. и x86, и ARM процессоры разбивают инструкции перед исполнением на микрооперации, чтобы исполнить мелкие команды наиболее эффективных для них способом, но x86 делает это при помощи микрокода процессора, а ARM это делает отдельным аппаратным средством, что эффективнее.

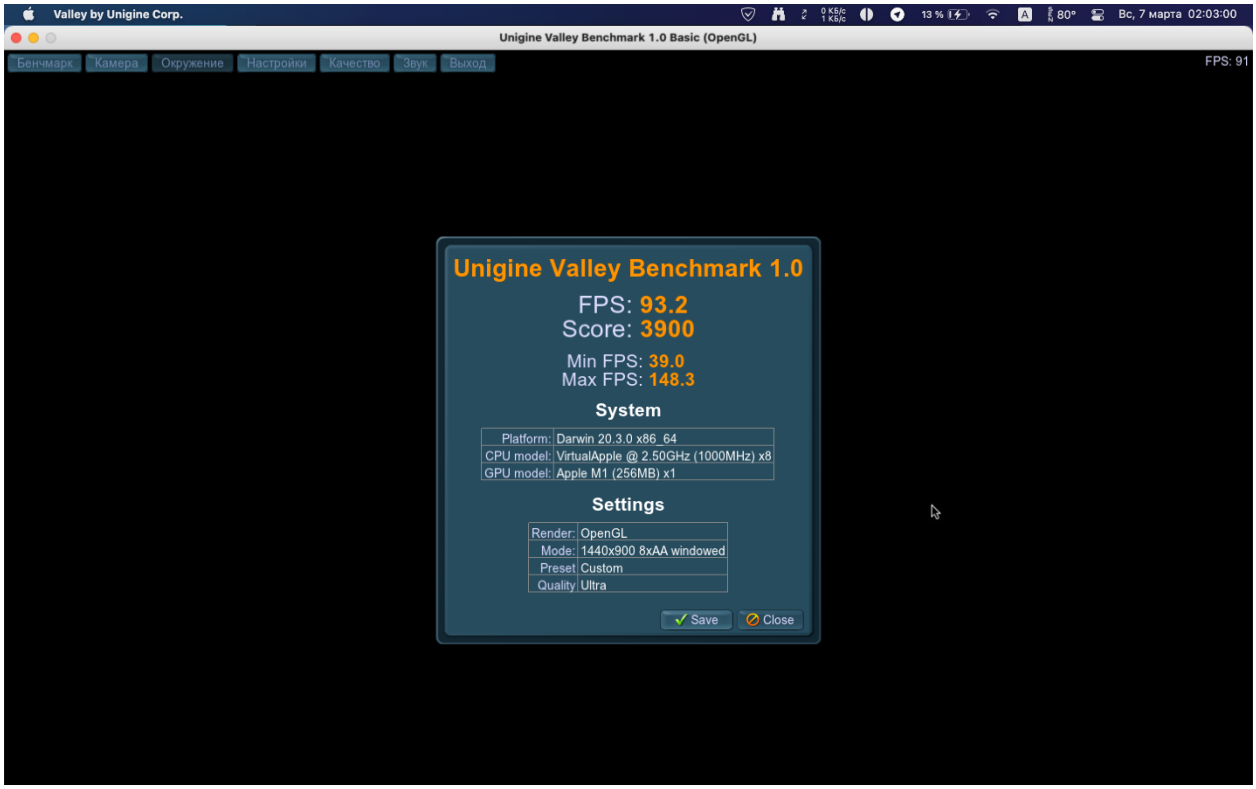
Почему бы просто не делать настоящие процессоры с широкими командами? Такие CPU не эффективны, выделяют многим больше тепла при прочих равных, поэтому x86 и ARM разбивают команды на микрооперации.

Если Вам привлекла тема устройства процессора, рекомендую видео Этот Компьютер, про AVX можете ещё узнать в видео Izzy Laif, Беларуский АЙТИшник, и статьях Huawei на habr и Dr. Ian Cutress на Anandtech.

2.4. Какие ещё есть ограничения у Mac на Apple Silicon?

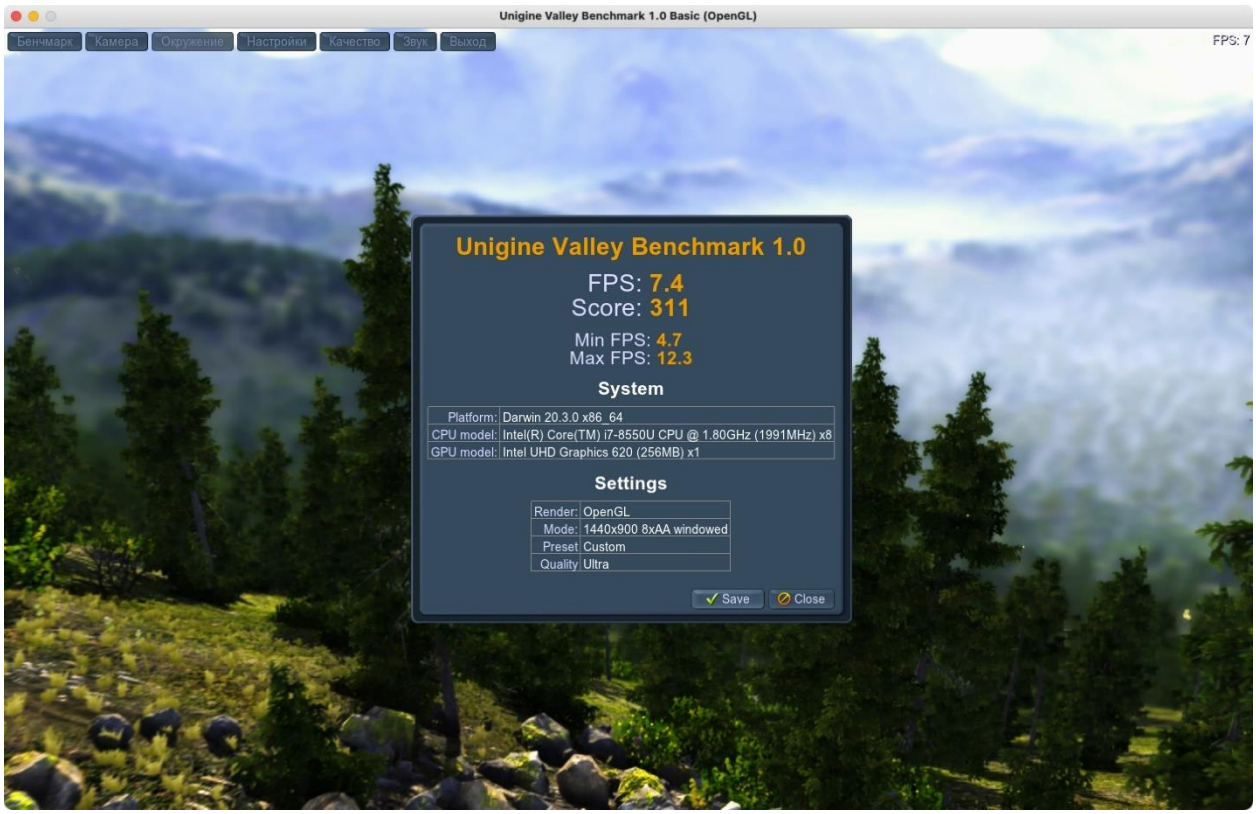
Остальные ограничения те же, что и у x86. И гарантии те же, что и у x86 по части остальных задач, также как и отсутствие гарантий в определённых задачах. К чему это я и какие гарантии отсутствуют у Apple?

И как этим объяснить проблемы с тестом Unigine Valley, о которых Роман Ястреб, которого я упоминал выше, сообщал в своём видео на 38:12?



Скриншот 17. Macbook Pro 13 2020 (Apple M1). Бенчмарк Unigine Valley, 900p, Ultra качество, сглаживание x8.

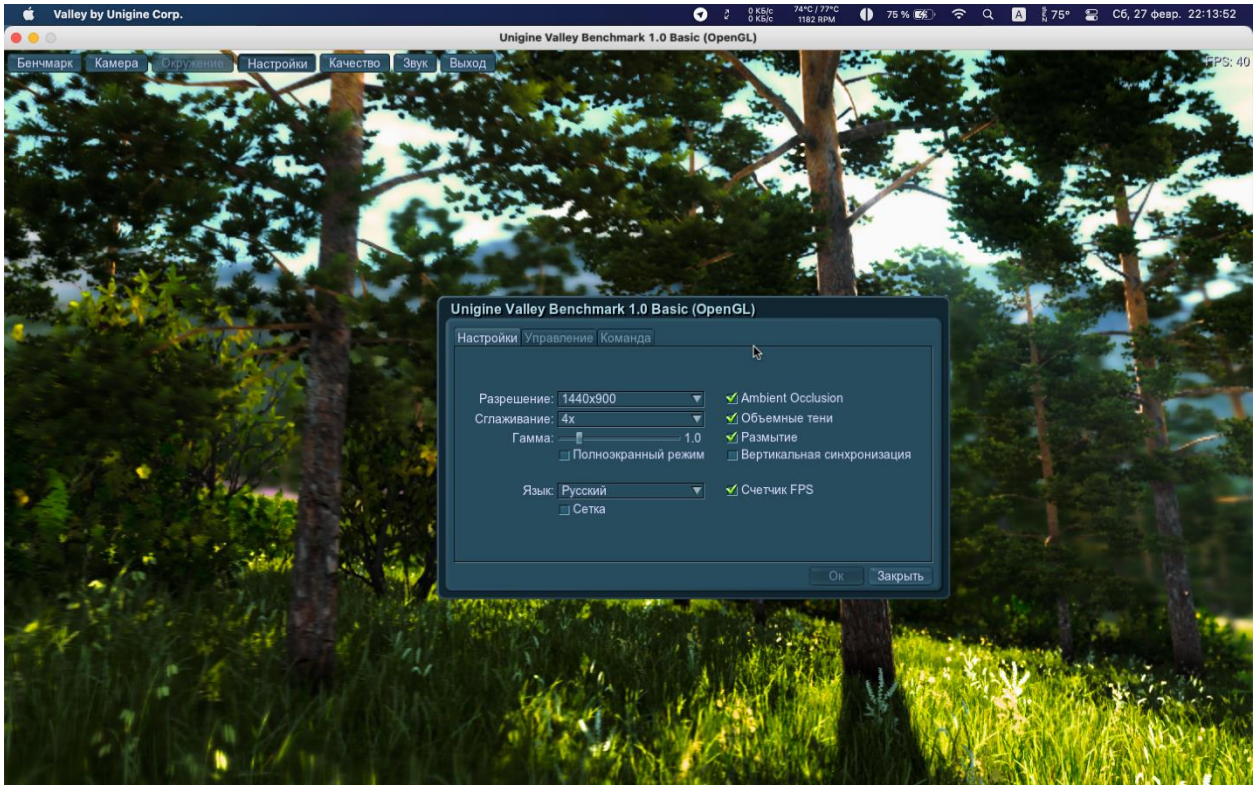
Да, действительно, на Macbook с M1 просто чёрный экран вместо реальной отрисовки сцен в Unigine Valley. Проверим это же на моём Xiaomi.



Скриншот 18. Xiaomi Mi Notebook Pro GTX. Бенчмарк Unigine Valley, 900p, Ultra качество, сглаживание x8.

Да, как Вы видите, действительно, у M1 есть проблемы с некоторым софтом, которые не присущи аппаратам на Intel с графикой от Intel/AMD.

Т.е. всё же M1 косячный? Вовсе нет.



Скриншот 19. Macbook Pro 13 2020 (Apple M1). Бенчмарк Unigine Valley, 900p, Ultra качество, сглаживание x4.

Если мы снизим сглаживание с x8 до x4 в том же Unigine Valley, то отрисовка картинки пойдёт как надо. Погодите-ка, но разве это не означает, что M1 всё-таки косячный или что Операционная Система macOS в этом плане имеет косяки? Почему же на x8 ничего не работает, а с x4 всё хорошо? Может быть, мне именно x8 сглаживание и требуется, а Apple мне его не позволяет задействовать, хотя было же обещано, что всё будет работать как надо. Или нет? На этот вопрос у меня есть простой ответ: Unigine Valley задействует для отрисовки API OpenGL. Это был короткий ответ.

Теперь более подробно: что же не так с API OpenGL? Дело в том, что Apple объявили API OpenGL и API OpenCL устаревшими и отказалась их поддерживать ещё в 2018 году с выпуском macOS 10.14 Mojave. Да, API OpenGL и API OpenCL не убрали полностью из системы, Вы всё ещё можете их задействовать, но Apple не несёт ответственности за исправную работу Вашего ПО с API OpenGL и API OpenCL. Т.е. позиция Apple, выражаясь более простым языком, примерно такая: «Вы можете использовать API OpenGL и API OpenCL, но никаких гарантий насчёт того, что Операционная Система поведёт себя с ними, как и задумано, мы не дадим, так что пинайте разработчика Вашего прикладного ПО, чтобы он переписал софт под Metal».

Проблемы с OpenGL могут возникнуть далеко не только в приложении Unigine Valley, но и в любом другом приложении. Приведу ниже пример.

Мне в определённый момент стало интересно проверить, как тут идут эмуляторы старых консолей через Rosetta 2, ведь под ARM их ещё не адаптировали. Я выбрал для проверки эмулятор Dolphin с открытым исходным кодом (эмулирует Nintendo GameCube и Nintendo Wii), его же я использую на своём старом ноутбуке Asus Q500A в Windows 10, когда хочется поностальгировать и поиграть с братом или сестрой в старый добрый Naruto Shippuuden Gekitou Ninja Taisen Special. Собственно, больше я ни в одну другую игру для Nintendo Wii или Nintendo GameCube не играю.

Dolphin ещё в 2018-ом году реализовал поддержку Vulkan на macOS через MoltenVK, транслирующий код для вызова API Vulkan в API Metal.

Почему бы прямо не задействовать Vulkan на macOS и зачем транслировать вызовы Vulkan в вызовы Metal? Я чуть подробнее раскрою этот момент в главе 2.6, так что надолго тут останавливаться не будем. Главное, что примечательно в Dolphin и почему его пример показателен: он имеет поддержку как нативного OpenGL, поддержка которого в macOS уже 2 года как прекращена, так и не нативного Vulkan, но более-менее толково работающего через трансляцию в API Metal.

Я заснял видео с экрана через QuickTime с геймплеем. И да, постоянно случаются подёргивания и игра регулярно подфризывает, причём с любыми настройками графики. Вы можете в этом сами убедиться, обратив внимание на значение FPS в правом верхнем углу Dolphin, и заметив, что FPS регулярно падает с 60 до 40-45 и именно в этот момент и становятся заметны фризы. Но, что занимательно, в видео на 10:55 после того, как я переключаю API видеокарты с Vulkan на OpenGL и включаю сглаживание, программа вылетает, т.е. она со сглаживанием в принципе отказывается работать с OpenGL, хотя с Vulkan никаких проблем не было. Без сглаживания между OpenGL и Vulkan, работающим через MoltenVK, особой разницы нет.

Плохо ли это, что API OpenGL и API OpenCL не поддерживаются и с ними могут возникать такого рода казусы, что на одном железе софт с ними работает как положено, а на другом с той же версией ОС ничего не работает? Я считаю, что да. Но можно ли компанию винить в том, что софт не работает, учитывая, что ранее за годы до этого она заявила, что никаких гарантий она не даст, приняв позицию «может, будет работать, а может – нет, ничего не знаем, лучше перепиши это дело под API Metal»? Я думаю, что нет.

Если Ваше ПО, которое Вам требуется в работе, всё ещё задействует API OpenGL или API OpenCL, не только не рассматривайте к покупке новые Macbook Pro или Air с чипом M1, но и старые Macbook не обновляйте даже до macOS 10.14, и уж тем более не обновляйте их до macOS 11.2 Big Sur.

Но зачем в принципе Apple отказались от поддержки API OpenGL и API OpenCL? Это всего лишь предположение (прошу не воспринимать мои домыслы как единственно верный ответ), но я думаю, что причина в

сложности обеспечить обратную совместимость всего старого ПО под Intel с новым Apple Silicon. Было бы очень дорого ещё и поддерживать технологии, и обеспечивать с ними обратную совместимость, потому, полагаю, и решено было от них отказаться. Точно также сложность разработки Rosetta 2 была бы несоизмеримо Выше, если бы пришлось ещё и обеспечивать обратную совместимость не только с 64-битными x86 приложениями (x86_64), но и с 32-битными x86 приложениями (x86), потому ещё в начале 2018 года с релизом macOS 10.13.4 High Sierra пользователи при запуске 32-битных x86 приложений стали получать оповещения о прекращении поддержки x86 приложений в macOS, начиная с 10.13.4. Т.е. ещё раньше, до релиза 10.14 Mojave, пользователи с x86 пребывали в точно таком же подвешенном состоянии «может, будет работать, может – нет, гарантий не дадим, лучше попросите разработчика Вашего ПО перекомпилировать приложение под x86_64», это не первый такой раз. И с релизом macOS 10.15 Catalina запуск 32-битных x86 приложений стал вовсе невозможен без виртуализации/эмуляции/трансляции при помощи сторонних решений.

Таким образом, ещё до релиза MacBook с чипом M1 на базе архитектуры ARM, Apple значительно упростили себе задачу по разработке Rosetta 2, сняв с себя обязательства по поддержке устаревших технологий, и к релизу macOS 11 многие разработчики уже выпустили 64-битные x86 приложения под macOS 10.15, которая 32-битные приложения уже не поддерживала, и уже немалое число разработчиков актуального ПО переписали своё ПО под API Metal, если ранее они задействовали API OpenGL или API OpenCL. Вот так они уже давно готовились к ARM.

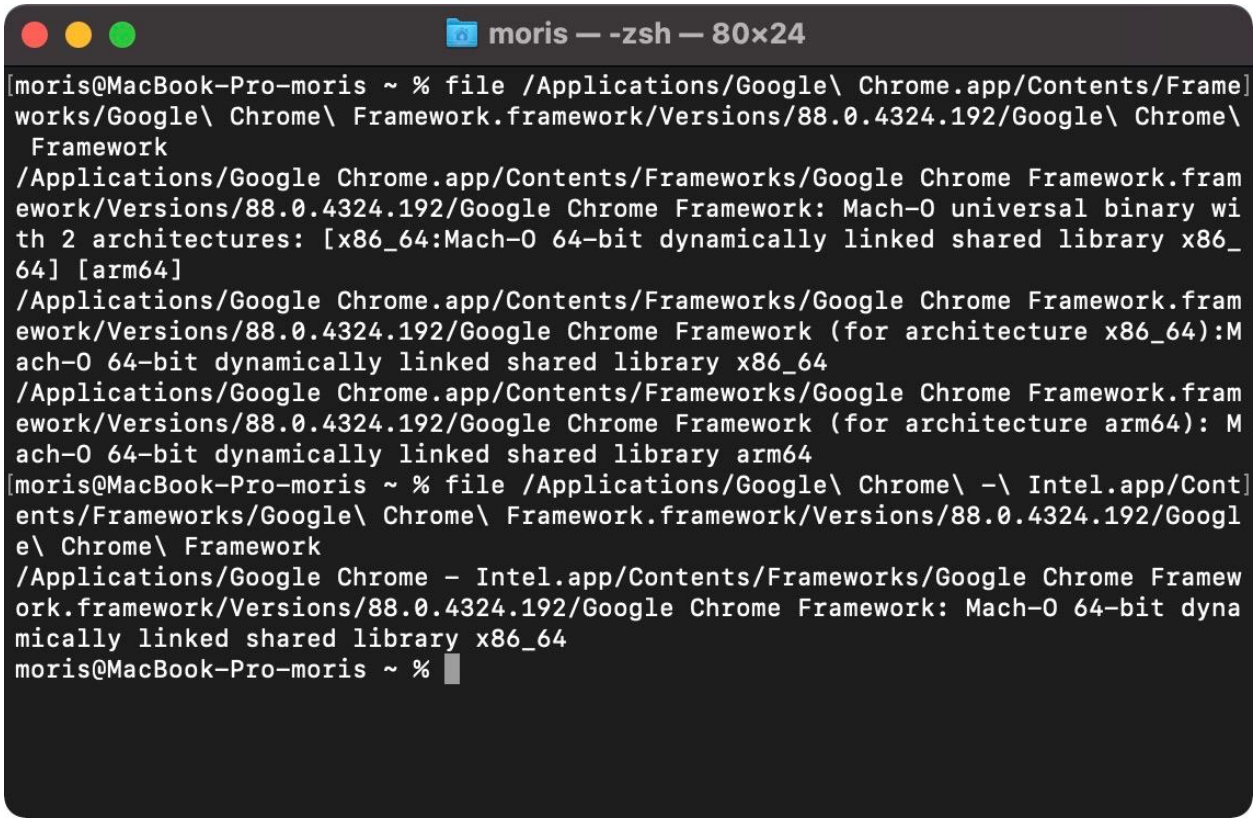
Unigine Valley всё ещё использует API OpenGL, потому не стоит удивляться тому, что с ним есть проблемы. С OpenGL гарантий нет и не будет, как и с OpenCL. Если они Вам нужны, не обновляйте Mac и macOS.

2.5. Зачем нужна Rosetta 2 для универсальных приложений?

На 34:18 Роман Ястреб в обзоре Macbook Air и Pro на M1 сообщил о проблемах с плагинами для Final Cut, которые ещё не поддерживают компьютеры под управлением macOS с архитектурой ARM.

Решение простое: запустить Final Cut через Rosetta 2 и продолжать использовать старые плагины. Дело в том, что Вы не можете просто «подружить» программы, написанные под Intel, с программами, написанными под Apple Silicon. Программы могут друг с другом полноценно корректно взаимодействовать только когда они написаны для одной и той же архитектуры. Но как же быть с программами под ARM и плагинами под x86?

Ну, тут всё очень даже интересно. Apple после релиза macOS 11 стала призывать разработчиков к разработке универсальных приложений, т.е. тех, которые имеют в своём исполняемом файле как скомпилированный машинный код для работы под x86, так и код для работы под ARM.



Скриншот 20. Информация о поддерживаемых архитектур браузером Google Chrome через Terminal.

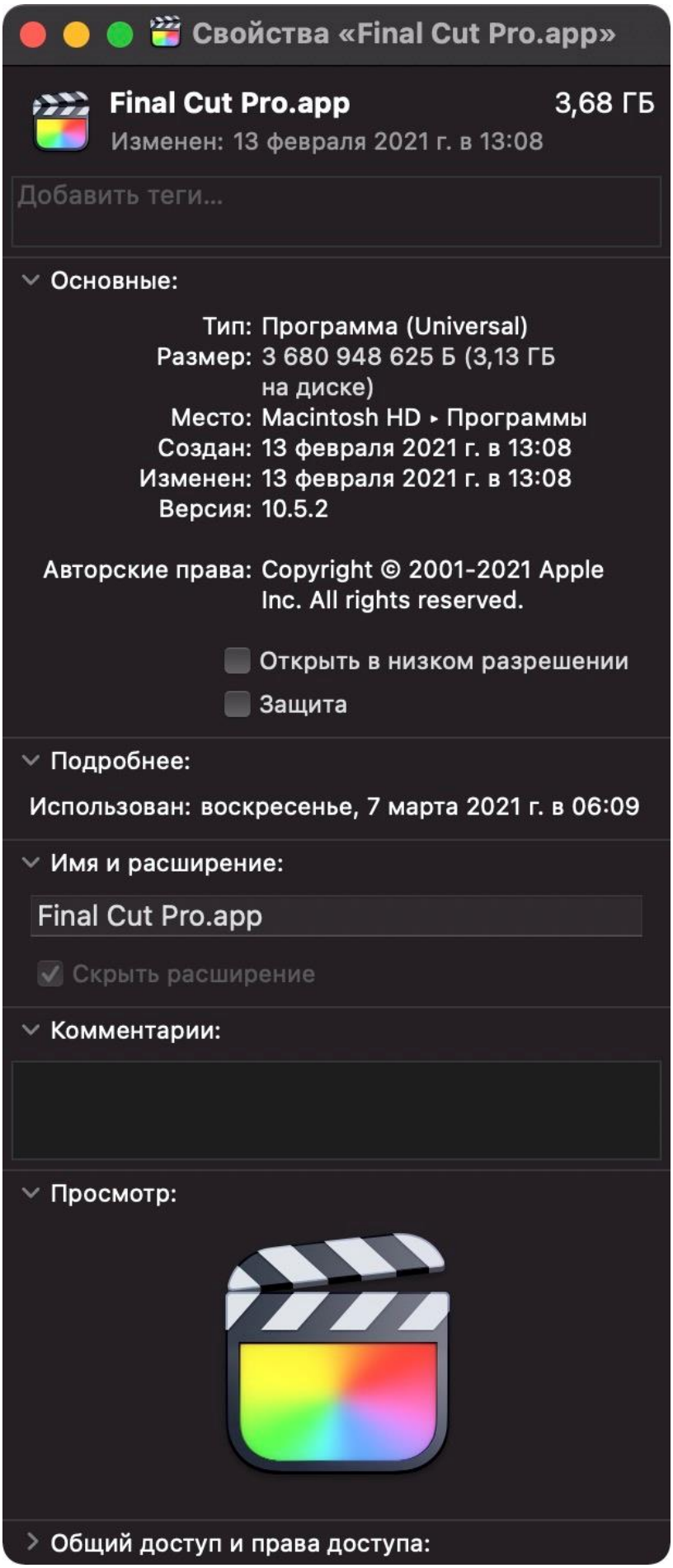
Пример: выше при помощи команды file путь_к_исполняемому_файлу я проверил, какие архитектуры поддерживаются браузером Google Chrome.

Как видите, версия Google Chrome для Intel имеет в своём исполняемом (бинарном) файле инструкции только для архитектуры x86_64, т.е. на Macbook с M1 её можно запустить только с помощью транслятора Rosetta 2.

В то же время Chrome для ARM включает в себя исполняемый файл (бинарный файл) с инструкциями сразу для обеих архитектур: x86_64 и arm64. Это значит, что Вы можете запустить Chrome как нативно на arm64, так и через Rosetta 2, предварительно транслируя часть машинного кода с x86_64 инструкциями в аналогичные arm64 инструкции, и тогда опыт Вашей работы с M1 в Chrome не должен отличаться от аналогичного опыта с Intel. Хотя, я что с Rosetta 2, что без, разницы не заметил в случае с Chrome. Если не обращать внимания на результаты бенчмарков, то в целом при повседневном использовании там разницы особой нет.

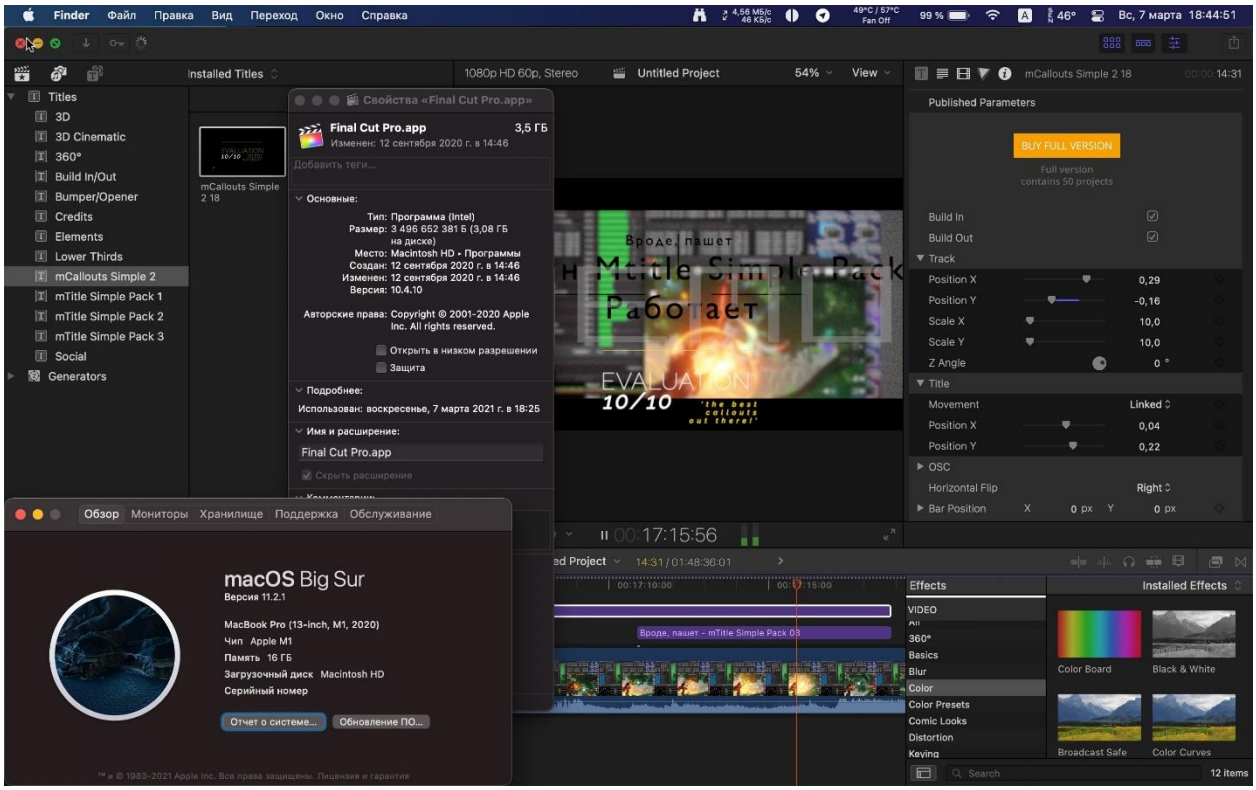
Вы можете запустить и приложение Final Cut с помощью Rosetta 2. В таком случае будет транслирован в arm64 машинный код, который отвечает за работу на x86_64 машинах. x86_64 версия пройдёт трансляцию кода через Rosetta 2 также, как её проходят Ваши расширения, и в результате они будут совместимы друг с другом во время работы. Да, так приложение Final Cut будет работать несколько медленнее, но как, я считаю, вполне себе сойдёт за временное решение. Как только надобность в Rosetta 2 в случае с данным приложением и его расширениями отпадёт, вы её спокойно выключите.

Погодите, как же запустить Final Cut через Rosetta 2, если опции Rosetta в случае с ним нет?



Скриншот 21. Macbook Pro 13 2020 (Apple M1). Информация о приложении Final Cut Pro X. Версия 10.5.2. Universal.

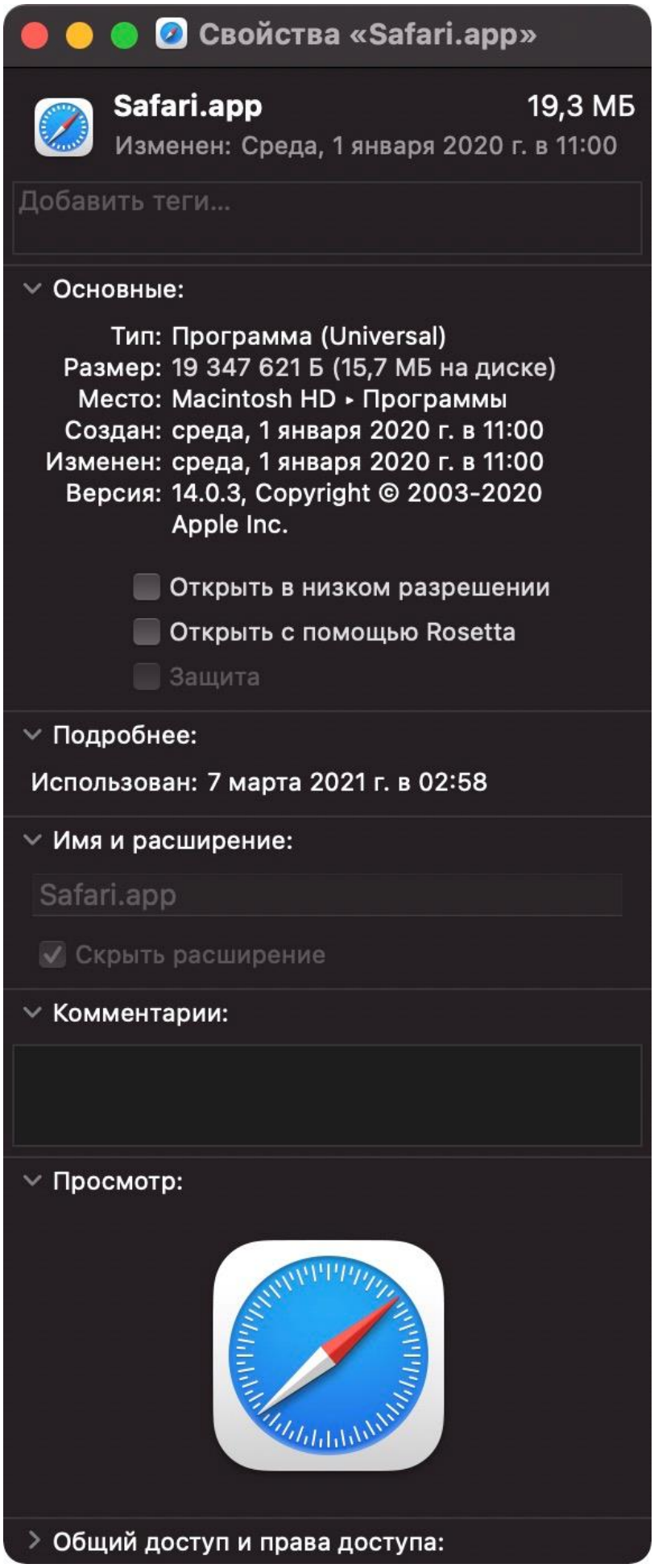
Да, действительно, новую версию Final Cut Pro X 10.5.2 вы не сможете запустить в режиме трансляции через Rosetta 2, и я не знаю, почему Apple наложили такого рода ограничение. А вот со старой версией всё в порядке.



Скриншот 22. Macbook Pro 13 2020 (Apple M1). Информация о приложении Final Cut Pro X. Версия 1.4.10. Intel only.

Как видите, старая версия, 10.4.10, которая предназначалась только для старых компьютеров Mac с SoC Intel, работает прекрасно через Rosetta 2 без каких-либо проблем. Учитывая то, что плагины под Final Cut писались под старую версию и их поддержка ограничена лишь этой версией, тем более имеет смысл использовать именно её, а не пытаться более свежую 10.5.2 завести на новых компьютерах Apple с ARM процессором в режиме трансляции через Rosetta 2. И со старыми версиями плагинов mCallouts Simple 2 и mTitle Simple Pack, первый из которых мелькал в видео Романа как проблемный, тут также нет никаких проблем. А новые версии плагинов имеют нативную поддержку как Final Cut Pro X 10.5.x, так и процессоров ARM, так что, как выйдут все необходимые обновления для Ваших плагинов, Вы потом и легко сможете обновить Final Cut до актуальной версии.

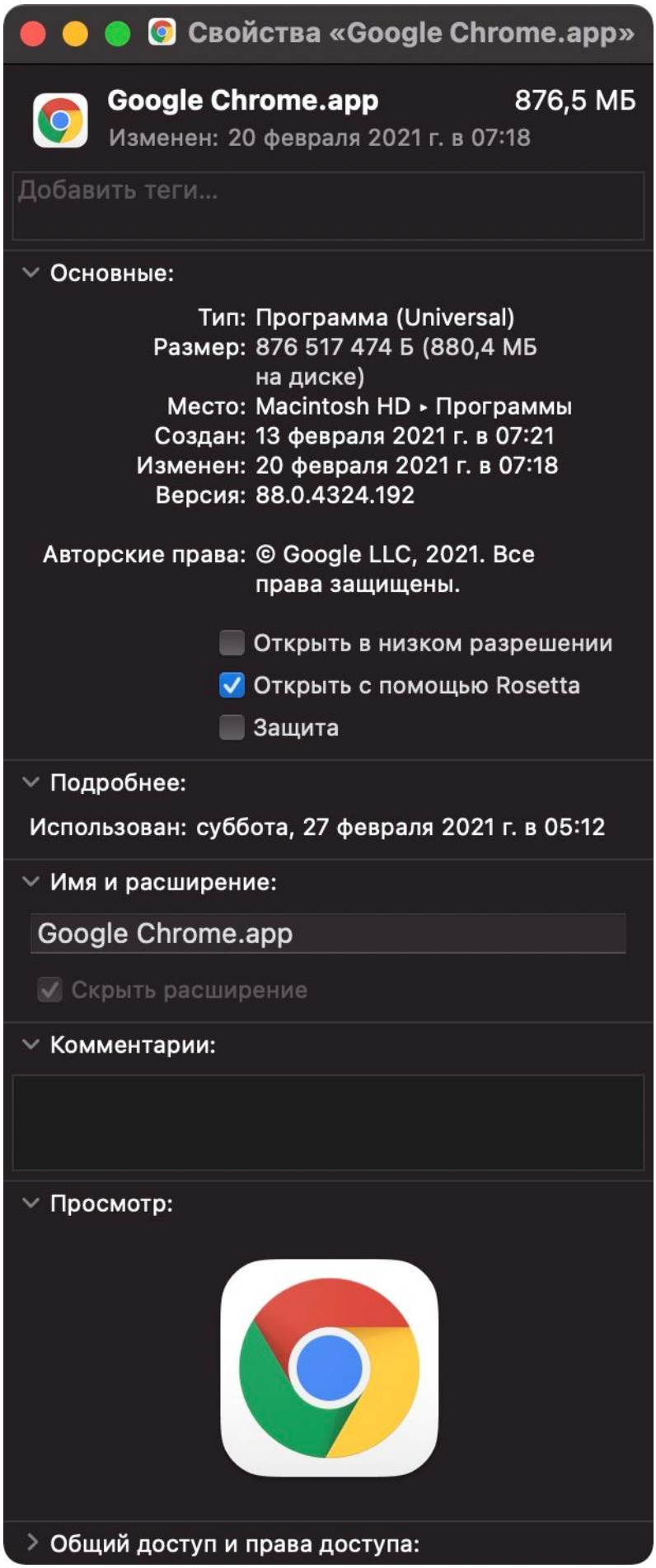
Опять же, для меня загадка то, почему Apple не разрешили запускать свежие версии Final Cut через Rosetta 2. С Safari ведь никаких проблем нет.



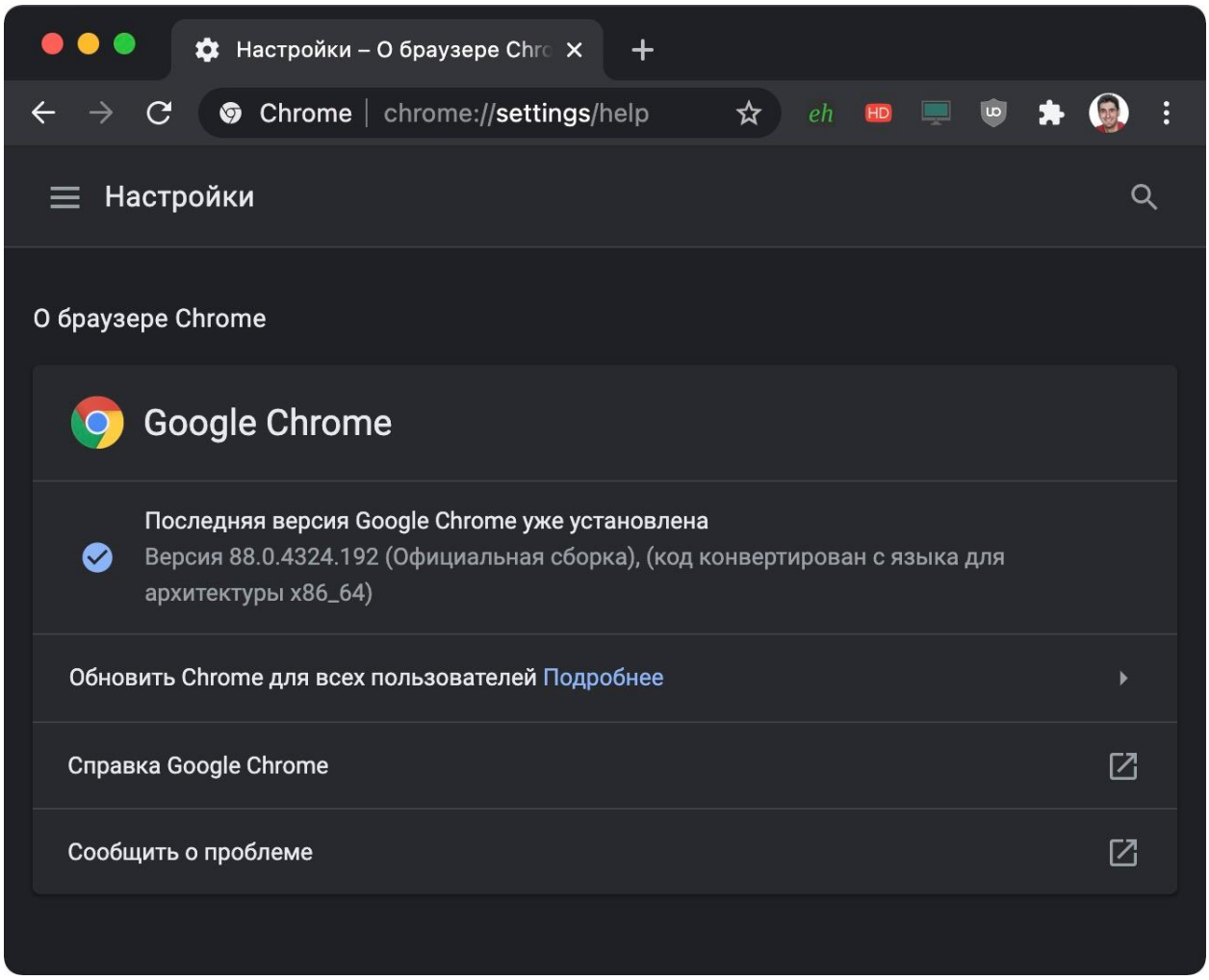
Скриншот 23. Macbook Pro 13 2020 (Apple M1). Информация о приложении Safari.

Если у Вас имеются какие-либо расширения для Safari, которые на данный момент ещё не работают с чипом M1, вы можете без труда запустить Safari с помощью Rosetta 2. В таком случае вместо использования нативного машинного кода под Apple Silicon, будет задействован транслированный код с x86 на ARM, и никаких проблем с поддержкой прежних расширений возникнуть по идее не должно.

Более того, Вы таким образом можете запустить с помощью Rosetta 2 любое универсальное приложение, в т.ч. и стороннее.



Скриншот 24. Macbook Pro 13 2020 (Apple M1). Информация о приложении Chrome.



Скриншот 25. Macbook Pro 13 2020 (Apple M1). Браузер Chrome. Страница "о браузере Chrome".

Я запустил через Rosetta 2 и браузер Google Chrome, и он работает, как и положено, никаких проблем в работе не заметил. Единственное, что дало знать, что Chrome работает через Rosetta, это текст «код сконвертирован с языка для архитектура x86_64» на странице «О браузере Chrome». К слову, точно такой же текст, как и при запуске через Rosetta 2 универсального бинарника, Вы увидите и при запуске через Rosetta 2 версии только для Intel.

Не знаю, какие расширения у Chrome требуют исключительно x86 архитектуру, т.к. все расширения из Chrome Web Store на Macbook Pro с чипом M1 без проблем заработали и без Rosetta 2. Chrome я показал исключительно примера ради, также как продемонстрировал, что и Final Cut старой версии без проблем функционирует через Rosetta 2 на новых компьютерах Apple с чипом M1. Есть проблемы с расширениями/плагинами? Запускайте основное приложение через Rosetta 2 и плагины к нему подтянутся также просто, как и на Mac с SoC Intel.

2.6. Неочевидные проблемы с macOS на M1 и их причины.

Странно, но почему-то все забывают, что для работы Java кода требуется виртуальная машина Java. Oracle официально ещё не выпустили JDK (Java Development Kit) и JRE (Java Runtime Environment) под компьютеры Apple с чипом M1, и, как мне кажется, причина тут проста: надо довольно много времени и сил потратить, чтобы переписать своё решение, ранее задействовавшее виртуализацию Intel, под новую технологию виртуализации Apple. Впрочем, компания Azul представила свою сборку OpenJDK, которая работает нативно на компьютерах Apple с чипами M1. Я загрузил их Zulu сборки OpenJDK 8 с их сайта и, вроде как, всё работает, но я не являюсь разработчиком на Java, у меня нет должных знаний в этой области, так что детальные проверки тут провести не смог.

К слову, про то, что построено на Java, да ещё и OpenGL: Minecraft. Роман Ястреб в своём ролике на 32:44 сообщил о проблемах с Minecraft при смене настроек графики. Ну, тут та же история, что и с Unigine Valley: «может, будет работать, может – нет, хотите гарантий – используйте Metal».

Также Роман Ястреб в своём видео на 33:36 рассказывает про проблемы с Dota 2 на Macbook с чипом M1. Этому также есть объяснение.

Dota 2 использует для отрисовки графики в игре либо OpenGL, либо Vulkan. С OpenGL ситуация понятна, а вот API Vulkan никогда и не поддерживался в macOS. Как же тогда он работает? Всё благодаря консорциуму Khronos Group, разработчику API OpenGL и Vulkan, и их разработке MoltenVK, которая транслирует вызовы API Vulkan в Вашем коде в аналогичные нативные вызовы в API Metal. Но озаботиться задействованием MoltenVK должен сам разработчик Вашего ПО. Вы, как пользователь, работая в macOS, не можете просто взять и задействовать в приложениях API Vulkan в чистом виде, никак не редактируя код этих приложений. Это именно задача разработчика «завернуть» Vulkan в Metal.

Dota 2 в своё время стала первым портом игры под macOS, задействующей трансляцию API Vulkan в API Metal через MoltenVK.

Вот только есть нюанс: MoltenVK SDK (Software Development Kit, инструмент для разработки) только с версии 1.2.162, вышедшей 9-го декабря 2020-го года, добавил поддержку формата пикселей, используемых GPU Apple в их Apple Silicon. И это ещё только начало. MoltenVK SDK в версии 1.2.170, исходный код которой был опубликован 23 февраля 2021 года, и готовый инсталлятор которой был представлен 26 февраля 2021 года на сайте LunarXchange, внедрил ещё одно исправление, благодаря которому GPU в Mac с чипом M1 начал корректно учитываться, как интегрированный графический ускоритель.

Роман выпустил свой ролик 30-го декабря и это нормально, что на тот момент разработка Dota 2 не переехала ещё даже на декабрьскую версию MoltenVK SDK, не говоря уже о февральском релизе, т.к. разработчики обычно трижды думают перед тем, как обновляться на новую версию их рабочего инструмента. Вы ведь тоже, надеюсь, не станете обновлять своё рабочее ПО, предварительно не удостоверившись, что оно исправно и никаких проблем с ним у Вас не возникнет? В добавок, мы не знаем, был ли API Vulkan задействован через MoltenVK в видео Романа во время игры в Dota 2. То вполне мог быть и старый API OpenGL, с которым есть проблемы.

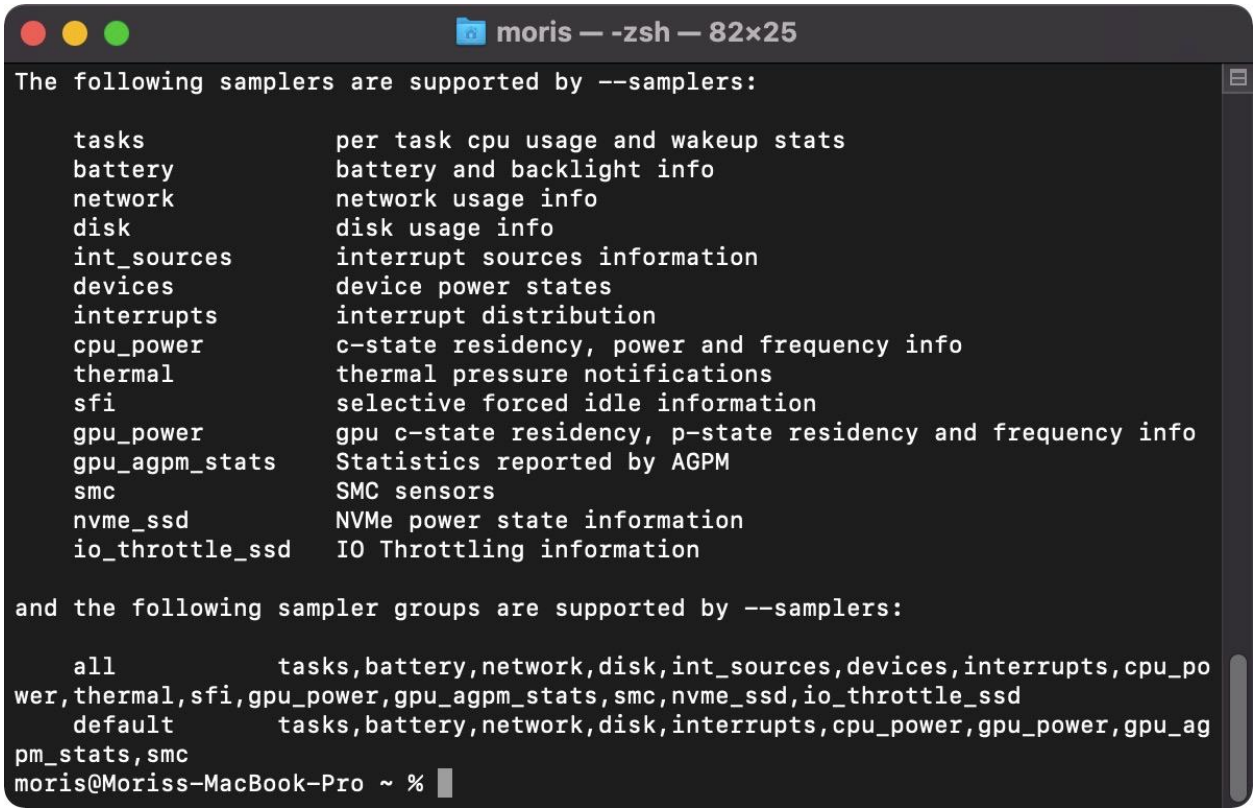
Сейчас, если разработчик использует последнюю версию MoltenVK, по идее, проблем быть не должно, но в идеале, конечно, лучше разработчику использовать напрямую API Metal, а не прописывать в коде своего приложения вызовы API Vulkan, чтобы затем транслировать API Vulkan в API Metal. Гарантии работоспособности ПО возможны только при использовании официальных SDK от Apple и при прямом использовании API Metal. Apple не причастна к разработке MoltenVK и винить компанию в проблемах, которые вызваны сторонними решениями, тем более их старыми версиями, как по мне, несколько некорректно.

А, если смотреть со стороны пользователя, имеется ли такая проблема? Да, имеется. Пофиксят ли это со временем? Скорее всего, да, т.к. уже сейчас видно, что разработчики активно решают обнаруженные проблемы, и скоро подобных проблем быть уже вовсе не должно. Но это не точно.

2.7. Какие есть отличия между macOS для x86 и ARM?

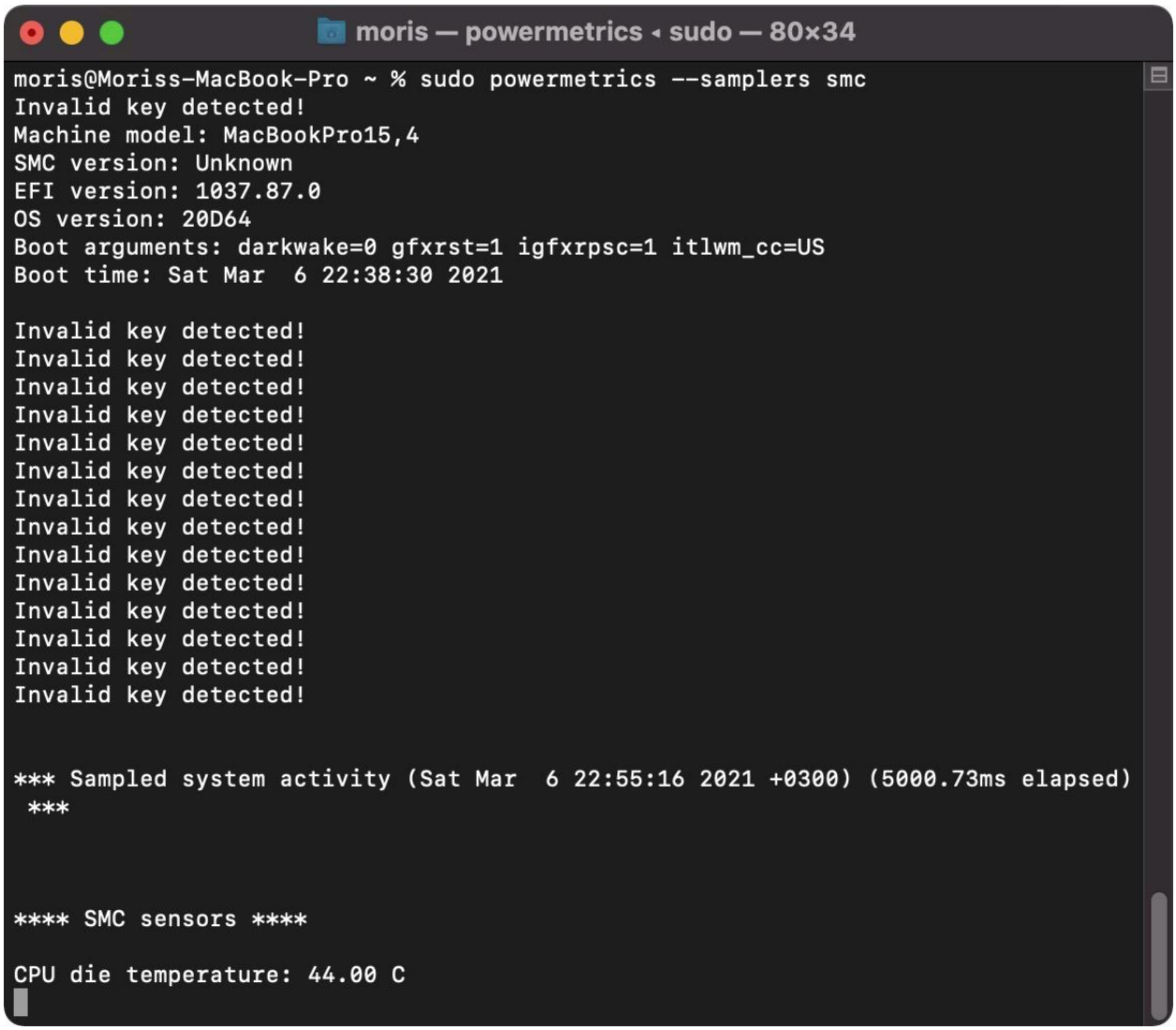
Вкратце: отличий дофига и больше. Начиная с того, что системные консольные утилиты выдают разные ответы на одни и те же команды, и заканчивая тем, что системные API возвращают абсолютно разные значения в ответ на одни и те же запросы.

Приведу пример. В macOS есть встроенная консольная утилита powermetrics, которая позволяет Вам получать в режиме реального времени информацию о состоянии Вашего оборудования. Таким образом можно мониторить температуру Вашего SoC, его энергопотребление / тепловыделение, его частоту, также можно мониторить и другие показатели.



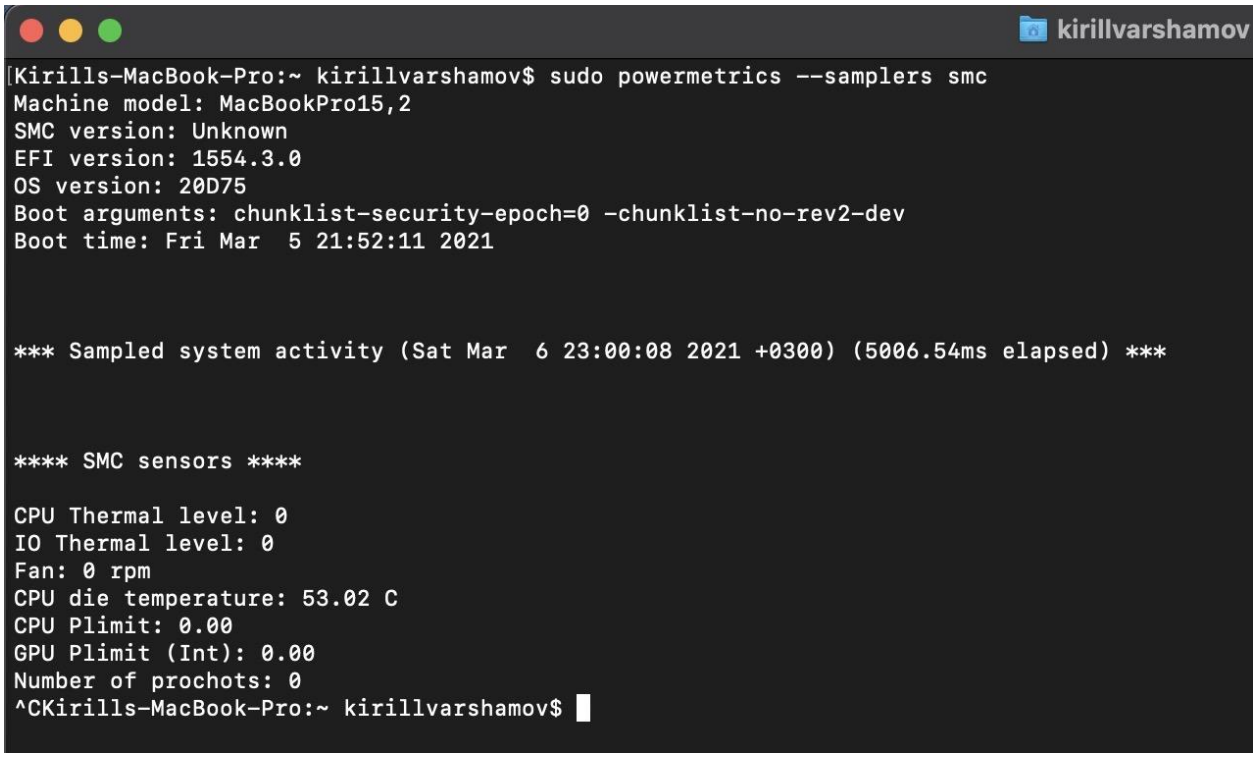
Скриншот 26. Xiaomi Mi Notebook Pro GTX. Информация из консольной утилиты Powermetrics.

Выше Вы видите вывод команды powersamplers –help. На ноутбуке Xiaomi она позволяет при помощи ключа –samplers smc отслеживать температуру. Попробуем.



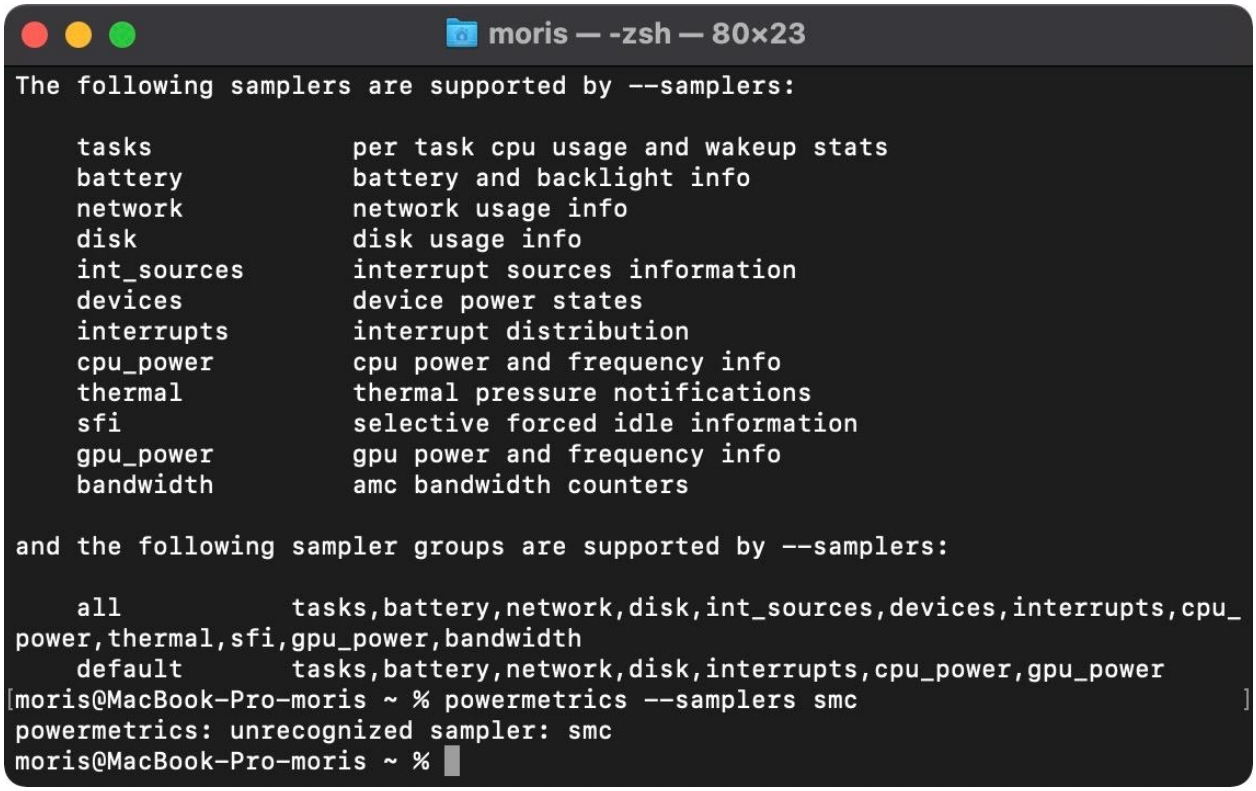
Скриншот 27. Xiaomi Mi Notebook Pro GTX. Информация из консольной утилиты Powermetrics (--samplers smc)

М-да. Утилита отобразила текущую температуру процессора, но заспамила нам Terminal сообщением “Invalid key detected”. Ну, тут уж куда деваться, это издержки хакинтоша. Опробуем эту же команду на MBP 2018.



Скриншот 28. Macbook Pro 13 2018 4xTB3. Информация из консольной утилиты Powermetrics (--samplers smc).

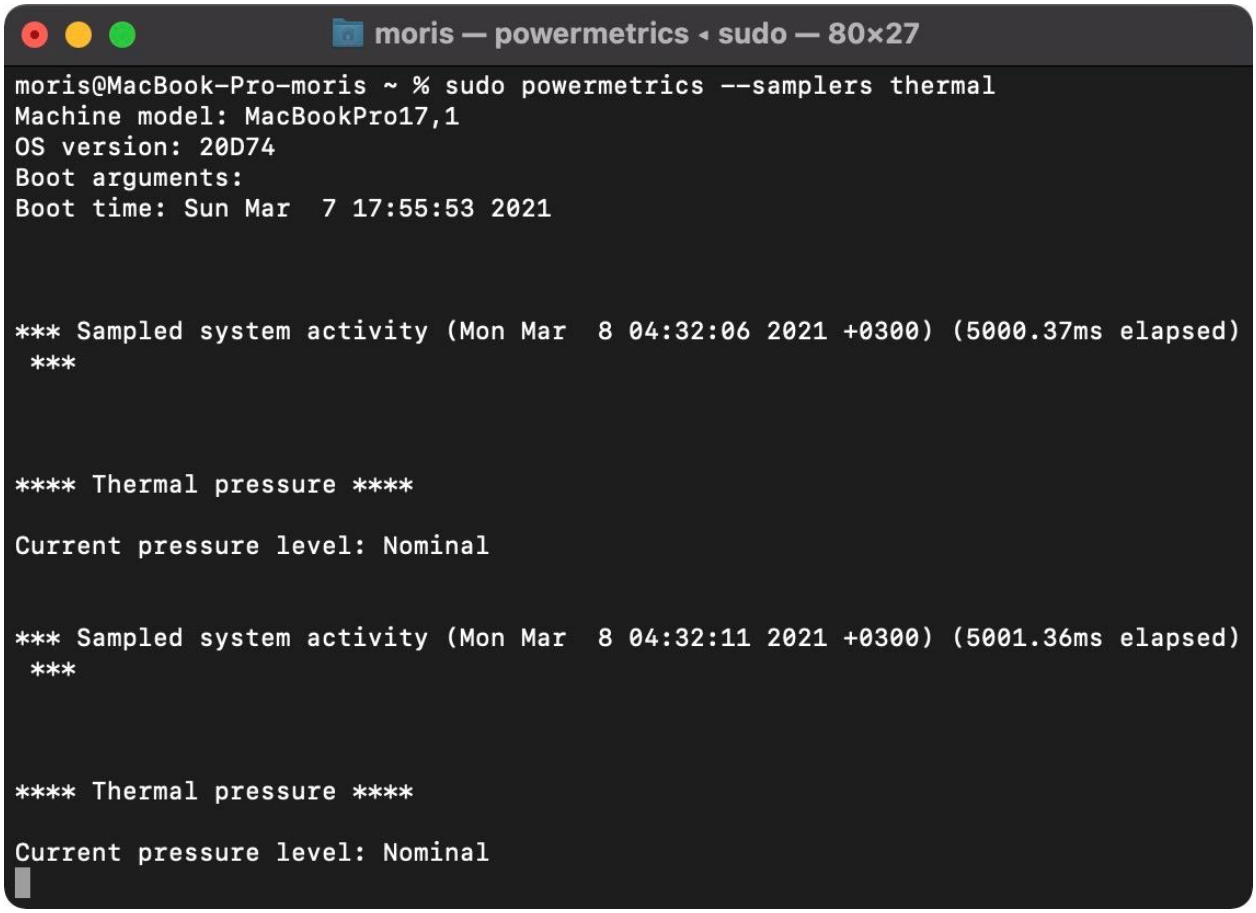
На настоящем Macbook Pro 13 2018 с 4x Thunderbolt 3 мы видим информацию о температуре процессора, скорости вентилятора, а также другие показатели. Тут понятно, теперь взглянем на Macbook с M1.



Скриншот 29. Macbook Pro 13 2018 4xTB3. Информация из консольной утилиты powermetrics (--samplers help).

Вот незадача. Утилита, которая на компьютерах Apple с SoC Intel на macOS 11 Big Sur позволяла проверить температуру процессора, уже не

позволяет сделать того же на компьютерах Apple с той же macOS Big Sur, но SoC Apple Silicon. У powermetrics попросту отсутствует smc в samplers, через который мы ранее запрашивали информацию. В целом, это логично, учитывая то, что и чип SMC в новых Macbook на Apple Silicon отсутствует, из-за чего нет необходимости сбрасывать NVRAM и SMC при возникновении проблем. Собственно, и проблем, из-за которых может понадобиться сбросить NVRAM и SMC более не возникает. Но ведь мы знаем, что у процессора M1 есть температурные датчики, как и любого другого процессора, и чип SMC на материнской плате тут не при чём. Наверняка, ответ в том, что нужно просто задействовать другую команду, например, powermetrics –samplers thermal. Попробуем.



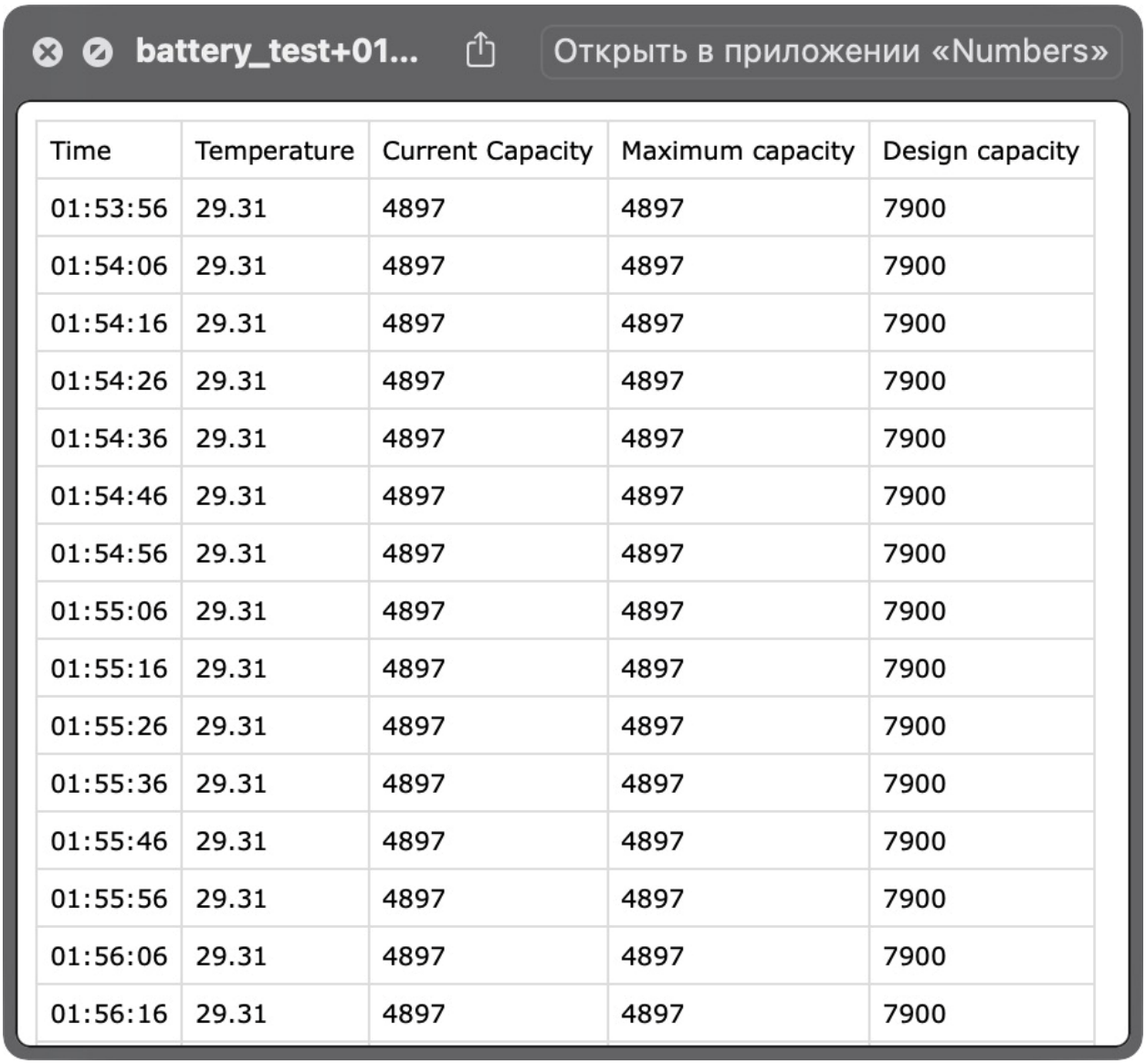
Скриншот 30. Macbook Pro 13 2018 4xTB3. Информация из консольной утилиты powermetrics (--samplers thermal).

Нет, и thermal sampler не возвращает нам температуры процессора. И даже более того скажу: ни один sampler этой информации не даёт.

Ну, ладно, скажете, что я докапываюсь, и кому вообще нужно брать информацию из Terminal, когда есть TG Pro и iStat Menus, которые по API получают нужную информацию из системы и корректно отображают

температуру чипа. В общем-то и с API есть проблема. Нет, не с API для запроса температуры процессора, но с API для запроса данных аккумулятора.

Мой хороший друг, Кирилл, написал на Swift (и немного Objective-C) утилиту для тестирования стабильности скорости SSD по мере заполнения. Так уж сложилось, что подобных утилит для macOS ранее никто не разрабатывал и тем более не публиковал с открытым исходным кодом, так что я ему благодарен за то, что взялся за это дело. Но также помимо того, что делает мой PowerShell-скрипт для заполнения SSD, он реализовал у себя в красивой обёртке и то, что делает мой PowerShell-скрипт для замера автономности ноутбуков, и ещё добавил возможность выводить отчёты по тесту SSD. Можете ознакомиться с исходным кодом его утилиты в его репозитории GitHub. Но к чему это я? Какое это всё имеет отношение к разнице в ответах по API, про которые я писал? У Кирилла в файле KABattery.m прописан запрос информации о состоянии батареи у AppleSmartBattery. Вот что выдаёт в ответ Xiaomi:

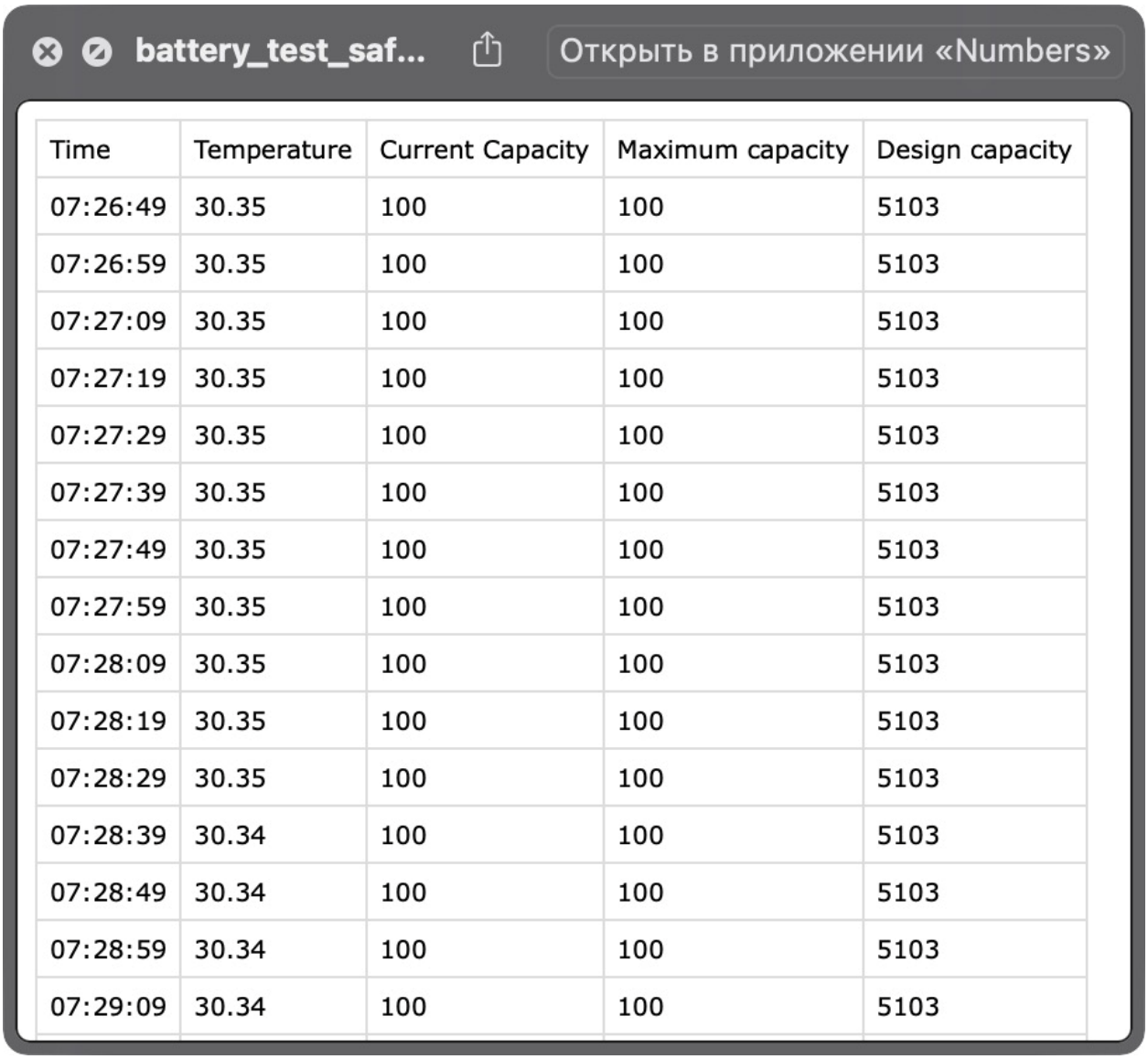


Time	Temperature	Current Capacity	Maximum capacity	Design capacity
01:53:56	29.31	4897	4897	7900
01:54:06	29.31	4897	4897	7900
01:54:16	29.31	4897	4897	7900
01:54:26	29.31	4897	4897	7900
01:54:36	29.31	4897	4897	7900
01:54:46	29.31	4897	4897	7900
01:54:56	29.31	4897	4897	7900
01:55:06	29.31	4897	4897	7900
01:55:16	29.31	4897	4897	7900
01:55:26	29.31	4897	4897	7900
01:55:36	29.31	4897	4897	7900
01:55:46	29.31	4897	4897	7900
01:55:56	29.31	4897	4897	7900
01:56:06	29.31	4897	4897	7900
01:56:16	29.31	4897	4897	7900

Скриншот 31. Xiaomi Mi Notebook Pro GTX. Таблица с результатом запроса данных у AppleSmartBattery.

Как видите, AppleSmartBattery на ноутбуке Xiaomi с процессором i7 8550U возвращает температуру батареи (фейковые данные, аккумулятор Xiaomi не передаёт в macOS реальной информации по температуре батареи), текущий заряд батареи в мАч (Current capacity), максимальный заряд батареи с учётом износа аккумулятора (Maximum capacity) и максимальный заряд, который должен быть у новой батареи с завода (Design capacity). Первое, на что хочется ругнуться, это на износ аккумулятора моего бл*дского Xiaomi на целых 38% за 2 года, но важно здесь не это. Важная деталь здесь в том, что и current capacity, и maximum capacity, и design capacity, которые были запрошены у AppleSmartBattery были выданы в мАч.

А теперь посмотрим, что покажет Macbook Pro на M1.

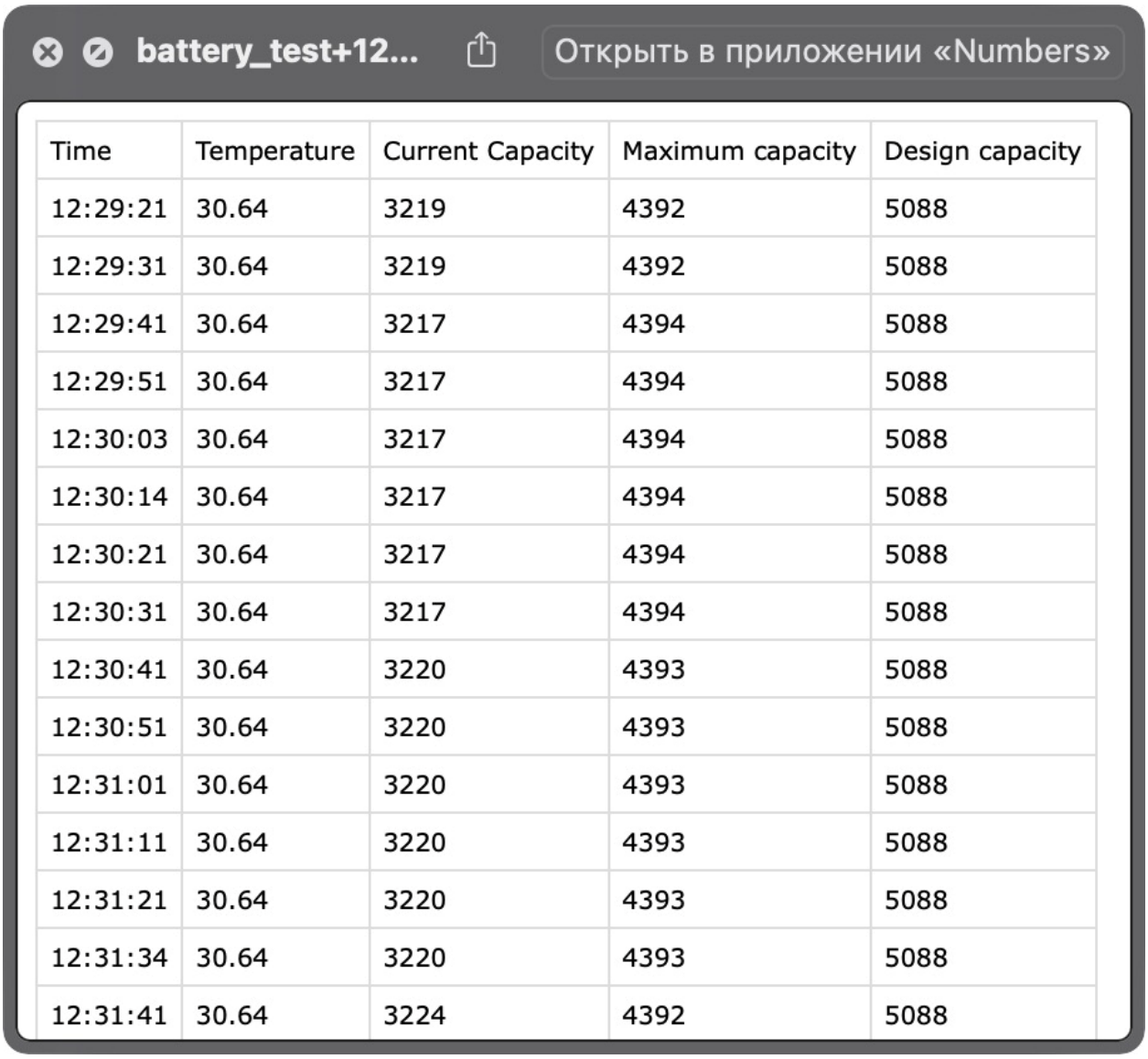


Time	Temperature	Current Capacity	Maximum capacity	Design capacity
07:26:49	30.35	100	100	5103
07:26:59	30.35	100	100	5103
07:27:09	30.35	100	100	5103
07:27:19	30.35	100	100	5103
07:27:29	30.35	100	100	5103
07:27:39	30.35	100	100	5103
07:27:49	30.35	100	100	5103
07:27:59	30.35	100	100	5103
07:28:09	30.35	100	100	5103
07:28:19	30.35	100	100	5103
07:28:29	30.35	100	100	5103
07:28:39	30.34	100	100	5103
07:28:49	30.34	100	100	5103
07:28:59	30.34	100	100	5103
07:29:09	30.34	100	100	5103

Скриншот 32. Macbook Pro 13 2020 (Apple M1). Таблица с результатом запроса данных у AppleSmartBattery.

Ну, температура тут настоящая, т.к. аккумулятор у Macbook в действительности имеет датчики температуры, и система корректно запрашивает данные, которые они фиксируют. И Design Capacity тут, как и положено, представлен в мАч, а по Current Capacity можно было бы решить, что ноутбук вот-вот разрядится, если бы не Maximum Capacity, который также составляет 100. Если вы подумали, что на Macbook на базе чипа Apple Silicon система возвращает вместо мАч информацию в % от заряда батареи, всё верно, вы угадали.

Не, может быть, это у меня просто хакинтош на Xiaomi кривой? Ну, раздул я тут из мухи слона, сейчас на Macbook Pro 13 2018 с чипом Core i5 8259U от Intel взглянем и всё встанет на свои места.



Time	Temperature	Current Capacity	Maximum capacity	Design capacity
12:29:21	30.64	3219	4392	5088
12:29:31	30.64	3219	4392	5088
12:29:41	30.64	3217	4394	5088
12:29:51	30.64	3217	4394	5088
12:30:03	30.64	3217	4394	5088
12:30:14	30.64	3217	4394	5088
12:30:21	30.64	3217	4394	5088
12:30:31	30.64	3217	4394	5088
12:30:41	30.64	3220	4393	5088
12:30:51	30.64	3220	4393	5088
12:31:01	30.64	3220	4393	5088
12:31:11	30.64	3220	4393	5088
12:31:21	30.64	3220	4393	5088
12:31:34	30.64	3220	4393	5088
12:31:41	30.64	3224	4392	5088

Скриншот 33. Macbook Pro 13 2018 (Core i5 8259U). Таблица с результатом запроса данных у AppleSmartBattery.

На Macbook Pro 13 2018 года с 4 портами Thunderbolt 3 мы видим точно такую же картину, как и на моём Xiaomi с хакинтошем. Разница только в том, что тут температура батареи настоящая, а не сфальсифицированная. Тут информация о текущем заряде батареи и максимальном заряде батареи также представлена в мАч, как и на моём Xiaomi. Ничего на места не встало.

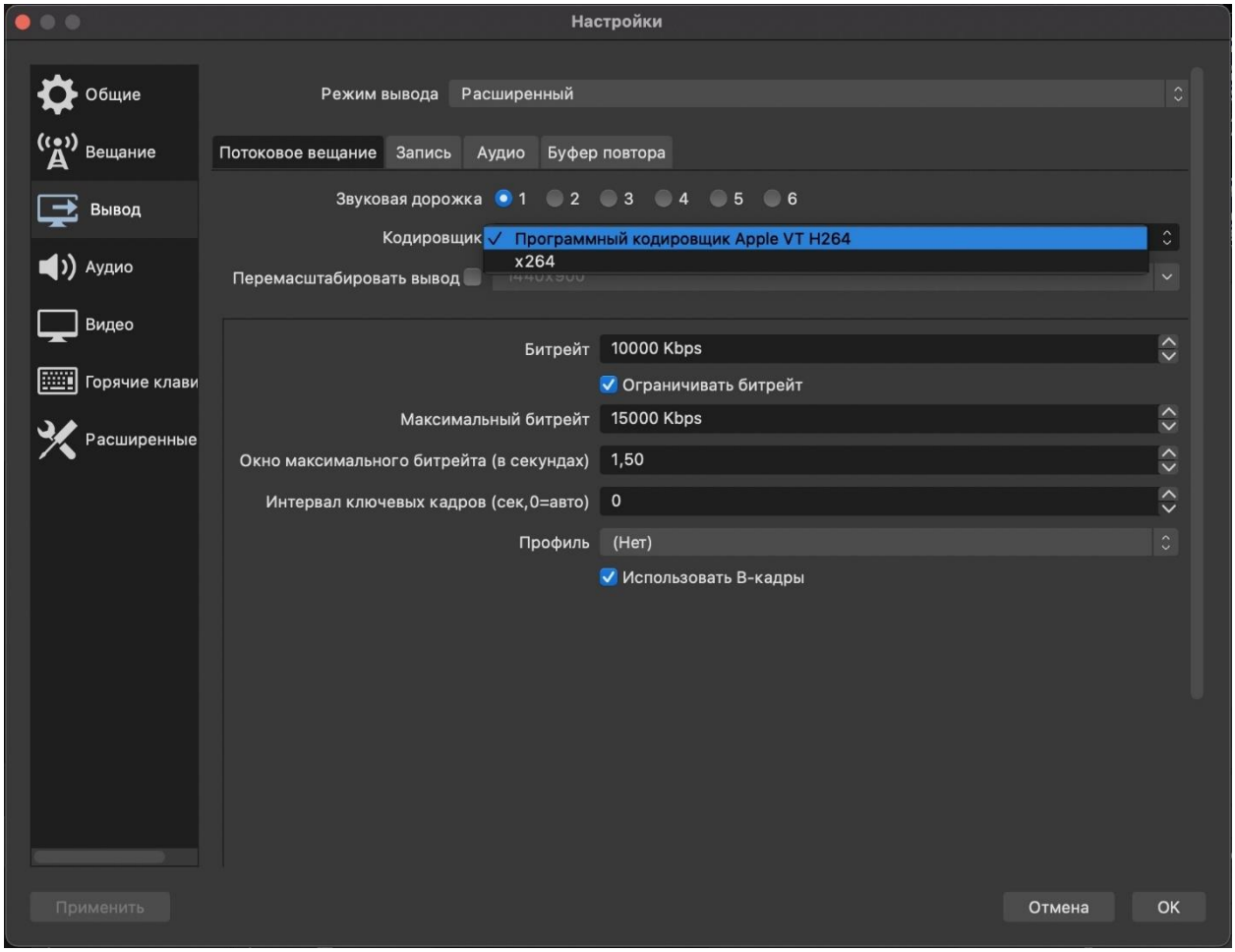
И тут возникает вопрос: Apple, а почему Ваша система на ноутбуках с Apple Silicon на те же запросы по API Вашему же AppleSmartBattery в ответ даёт разные данные в разных единицах измерения? Почему Операционная Система на запросы, касающиеся батареи, которая особо между ноутбуками не отличается, откликается по-разному? Это же не OpenGL, который Вы более не поддерживаете, это же Ваш же API, Ваш же AppleSmartBattery.

Может быть, проблема в том, что программа под Apple Silicon как-то криво скомпилировалась? Нет, я запускал и через Rosetta 2, всё то же самое.

Возможно, многие не сразу поймут, в чём же тут проблема, ведь есть другие API, по которым можно стянуть информацию о состоянии аккумулятора. Да, это так, но проблема в том, что Apple о подобной разнице в поведении Mac на Intel и Mac на Apple Silicon не оповещает. Допустим, Вы – разработчик, и Вы разработали приложение, в одной части которого проводится подсчёт % износа аккумулятора, и Вы для сбора данных об аккумуляторе опираетесь на AppleSmartBattery. Вы стягиваете с него информацию о максимальном заряде батареи (maximum capacity) в мАч и делите его на изначально задуманный объём аккумулятора (design capacity), а затем отнимаете от 100% это число. Предположим, у вас аккумулятор сейчас может иметь максимальную ёмкость 4500 мАч, а изначально аккумулятор должен был иметь ёмкость 5100 мАч. Мы делим 4500 на 5100 и получаем 0,88. Умножаем 0,88 на 100% и отнимаем от 100% полученное число (88%). В результате мы узнаём, что износ аккумулятора составил 12%. Всё просто, верно? Но вот в случае с Mac на Apple Silicon в maximum capacity всегда будет 100, и Вы вместо 4500 мАч на 5100 мАч делите 100 (% вместо мАч) на 5100. Выходит 2%. 100% - 2% = 98%. Ужас, у Вас аккумулятор изношен на 98%, как он работает то вообще? Пользователь, разумеется, решит, что программа врёт, т.к., как же так, ведь его Macbook работает X часов, почти столько же, сколько и работал первоначально. Он влепит Вам 1 звезду в App Store, а Вы, сидя за Macbook на Intel и видя, что программа то у Вас работает так, как и задумано, будете совершенно не понимать, в чём причина недовольства. Программа ведь у Вас успешно скомпилировалась под обе архитектуры, и у себя Вы её протестировали, и вообще всё прекрасно. В чём дело то тогда?

Это лишь 1 гипотетический пример того, какие проблемы может обеспечить Вам разный ответ ОС на одни и те же запросы в случае с разными ноутбуками. И я не понимаю, почему существует такая разница на одной ОС.

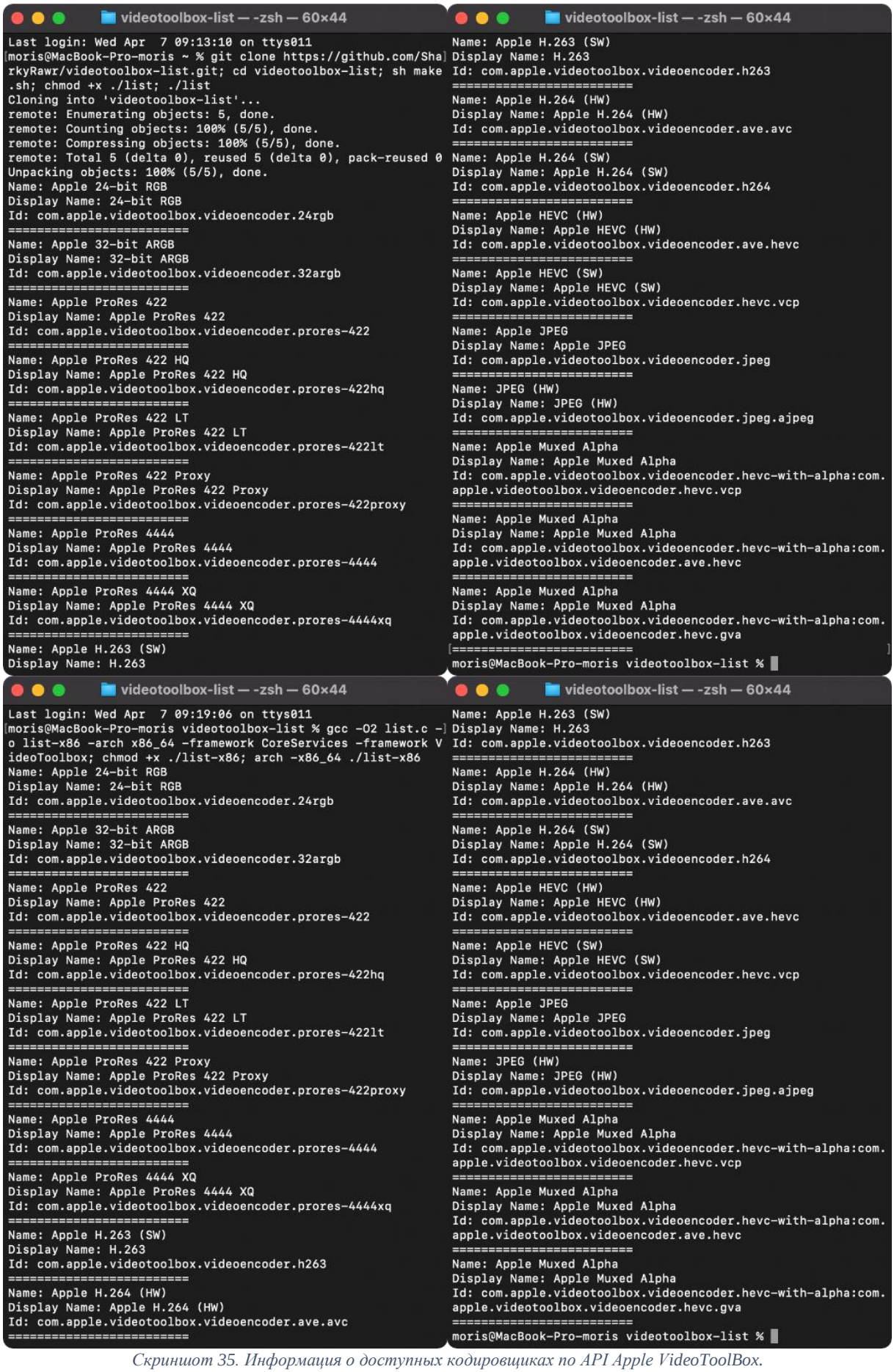
Есть ещё ряд нюансов, с которыми Вы можете столкнуться. И этот нюанс вызван, опять же, тем, что Apple даёт разные API для x86 и ARM.



Скриншот 34. Отсутствие аппаратного кодировщика Apple VT (VideoToolBox) H264 в OBS.

В разделе неполадок (issues) в репозитории OBS в GitHub пользователи новых компьютеров Mac с SoC Apple M1 открыли тему по поводу неработающего аппаратного ускорения кодирования видео в OBS. Оно не работает как для стримов, так и для записи. Поэтому вместо кодирования видео аппаратными блоками в SoC Apple M1, на данный момент из опций имеется только кодирование силами центрального процессора внутри SoC.

Оказалось, что всему виной переименованные компанией Apple названия кодировщиков видео (вернее, их ID). Пользователь SharkyRawr написал небольшую утилиту на C, которая запрашивает у ОС имеющиеся кодировщики видео и их ID, и выводит пользователю. И ответ ОС не зависит от того, скомпилирована ли утилита для x86_64 платформы или же для ARM, исполняется ли утилита нативно или же через предварительную трансляцию Rosetta 2. Ответ у Mac с M1 всегда тот же, и он не такой, как у Mac с Intel.



Скриншот 35. Информация о доступных кодировщиках по API Apple VideoToolBox.

Я проверил это у себя и действительно всё так и оказалось. Обратите внимание на значение ID у кодировщика Apple H.264 (HW): он в Mac с SoC

Intel назван **com.apple.videotoolbox.videoencoder.h264.gva**, а здесь же он имеет имя **com.apple.videotoolbox.videoencoder.ave.avc**. OBS обращается к первому, но его в ОС для Mac на Apple Silicon попросту нет, тут он назван иначе.

Apple, зачем вы так сделали? Apple VideoToolBox ведь по факту в macOS являлся ранее лишь надслойкой для Intel Quick Sync Video, так в чём была проблема обеспечить обратную совместимость этого же самого API с Вашим же новым железом, не называя кодировщик в новом SoC иначе? Тут никакая Rosetta 2 не поможет, если OBS вызывает то, чего не существует.

Поясню: дело в том, что в SoC Intel Core (и не только Core) с интегрированным графическим ускорителем уже лет 10 как имеются интегрированные внутри SoC отдельные вычислительные блоки для кодирования видео в формате H.264. Эти вычислительные блоки с годами обзаводятся поддержкой всё большего числа форматов, включая H.265, VP8, VP9 и AV1, но нас в данном контексте интересует именно кодек H.264. Они (блоки) выполняют задачу по кодированию видео намного эффективнее, чем это делает центральный процессор, тратя намного меньше энергии и выделяя намного меньше тепла. У них всего одна задача и они её выполняют хорошо.

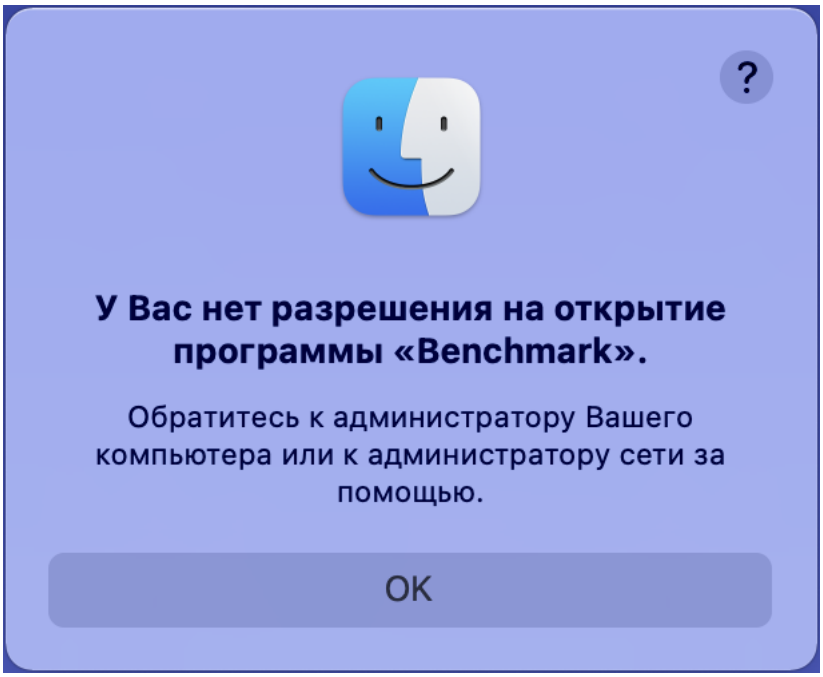
В Windows и Linux для задействования аппаратного кодирования ускорения этими блоками используется libmfx, либо Media Foundation в Windows и VAAPI в Linux. Прочитать об этом кратко можете в wiki ffmpeg.

В macOS же Apple задействовали свою реализацию, API VideoToolBox, для доступа к тем же аппаратным блокам Intel Quick Sync Video. Таким образом было упрощено взаимодействие программистов с блоками аппаратного кодирования и декодирования видео. Что там происходит на более низком уровне, программист мог вовсе не знать, он задавал выполнение определённой задачи при помощи API VideoToolBox, а система уже за него обращалась к блокам Quick Sync Video. Почему нельзя было реализовать то же, что и с FaceID в 2017 году? Тогда API остался тем же, что и для TouchID, просто система по нему выполняла уже другие действия.

Из-за этого разработчикам OBS требуется переписать код для аппаратного ускорения. Разный ответ по одним и тем же API... М-да...

Более того скажу, macOS ведёт себя непредсказуемо даже в рамках одной и той же модели компьютера с одной и той же SoC Apple M1.

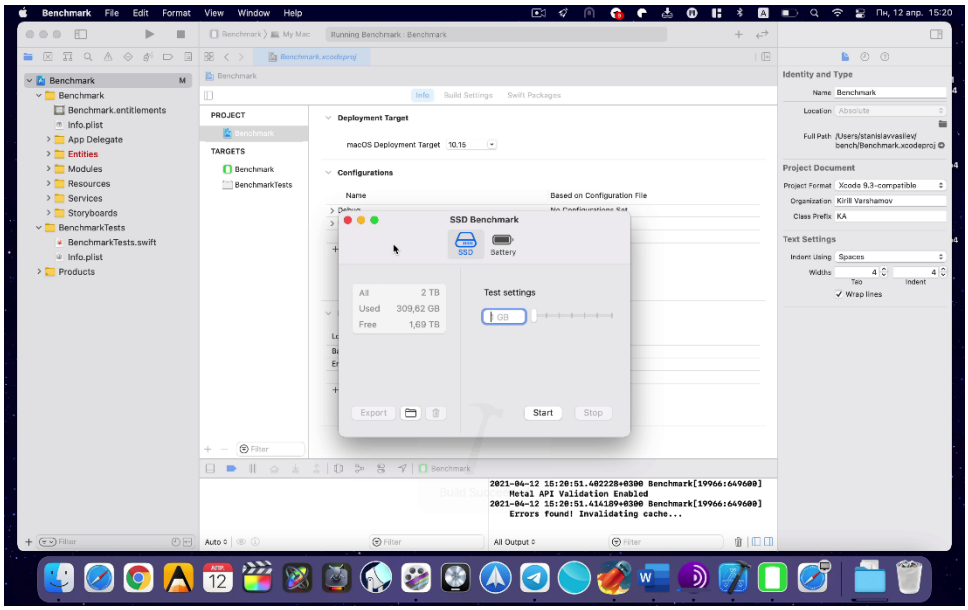
Я провёл тесты SSD на 256 GB модели Macbook Pro с чипом M1, который мне предоставили Цифрус, и мне захотелось сравнить поведение SSD на 256 GB и на 2 TB в Macbook Pro 13 с Apple M1, который на руках у Стаса. Мы со Стасом попробовали запустить тот же тест от Kavars, который успешно стартовал на моём экземпляре M1, но тут уж куда деваться.



Скриншот 36. Macbook Pro 13 2020 (Apple M1) на 2 ТБ (Стаса). Ошибка при запуске утилиты Benchmark от Kavars.

Мы, естественно, не копировали приложение с одного компьютера на другой, запаковывая приложение в .zip и ломая его права доступа, а установили приложение, перетянув его из .dmg контейнера в папку приложений. Мы также пробовали вручную задать права доступа и владельца через Terminal, но что бы мы ни делали, прав доступа мы не заимели, и в настройках в разделе безопасности также никакого запроса не возникло.

И знаете, что самое странное? Нам удалось запустить это приложение, на экземпляре Macbook Pro 13 с Apple M1 у Стаса после перекомпиляции его из исходного кода. Причина оказалась вовсе не в разрешениях на запуск.



Видео 1. Компиляция утилиты Benchmark от Kavars через Xode. Попытка запуска версий для “Any Mac” и "My Mac".

Если компилировать эту утилиту из исходного кода для «Any Mac», то на экземпляре Macbook Pro с Apple M1 у Стаса она не стартует, но при этом запускается на моём экземпляре Macbook Pro с тем же Apple M1 и стартует на моём Xiaomi с хакинтошем. Однако, если скомпилировать её под «My Mac» или «My Mac (Rosetta)», никаких проблем нет, всё работает как надо.

```
MacOS — -zsh — 80×24
Last login: Sun Apr 11 00:21:55 on ttys001
[stanislavvasilev@MacBook-Pro-Stanislav-2 ~ % cd Desktop/Benchmark\ 2021-04-12\ 1]
5-39-36/Benchmark.app/Contents/MacOS/
[stanislavvasilev@MacBook-Pro-Stanislav-2 MacOS % file Benchmark                  ]
Benchmark: Mach-O universal binary with 2 architectures: [x86_64:Mach-O 64-bit e
xecutable x86_64] [arm64:Mach-O 64-bit executable arm64]
Benchmark (for architecture x86_64):    Mach-O 64-bit executable x86_64
Benchmark (for architecture arm64):     Mach-O 64-bit executable arm64
stanislavvasilev@MacBook-Pro-Stanislav-2 MacOS %
```

Скриншот 37. Архитектуры, под которые имеются инструкции в утилите Benchmark, собранной для "My Mac".

Что вызывает ещё больше вопросов, так это то, что при компиляции утилиты Benchmark от Kavars (Кирилл Варшамов) под «My Mac», в получившемся исполняемом файле также имеются и x86 инструкции. Apple, а в чём тогда разница при компиляции под «My Mac» и под «Any Mac»?

Перед тем, как проверить, какие инструкции содержит в себе исполняемый файл приложения, я заархивировал его и экспортировал на рабочий стол, т.е. я проверял инструкции уже скомпилированного автономного приложения, не требующего запуска через XCode, а не дебаг-версии, которая запускалась бы только через XCode и никак иначе.

Т.е., неважно, компилируется ли утилита под «My Mac» или же «Any Mac», в итоге исполняемый файл можно запускать как на Mac с SoC Intel, так и на Mac с SoC Apple Silicon, но так лишь в теории. На практике на одних экземплярах Mac с Apple Silicon софт будет работать, будучи собранным под «Any Mac», а на других – нет.

Причём, наши со Стасом экземпляры отличаются только объёмом постоянной памяти: в обоих Macbook Pro установлен SoC Apple M1 с 16 GB объединённой памяти, и разница лишь в объёме SSD. У меня модель с 256 GB SSD, а у Стаса – 2 TB SSD. Всё, других отличий нет. Так почему же здесь имеется какая-то разница в том, какие приложения можно запускать на экземпляре Цифруса, и какие можно на экземпляре Стаса, который он приобретал чисто под своих нужды. Как так получилось то, Apple?

Вы можете ознакомиться с исходным кодом утилиты Кирилла в его репозитории GitHub и убедиться, что ничего специфичного там нет. И я снова не понимаю, что могло послужить причиной этого.

2 практически одинаковых компьютера, отличающихся лишь объёмом постоянной памяти, работают по-разному, т.е. одни и те же утилиты, собранные одним и тем же образом (Target – «Any Mac») на одном экземпляре запускаются, а на другом - нет. Почему нельзя быть уверенным в том, что одно и то же ПО, работающее на одном Macbook Pro с Apple M1, будет работать и на другом Macbook Pro с Apple M1? Что это за рулетка такая? У меня же ответа нет. Дичь. Я надеюсь, что представительство Apple прочитает эту статью или ознакомится с видео Стаса и прокомментирует этот момент хоть как-то. Хотелось бы знать причину такого поведения Macbook Pro на Apple M1, хотелось бы знать и способы решения подобных проблем.

2.8. Внешние видеокарты (eGPU) и USB4/Thunderbolt.

У новых Mac с SoC Apple Silicon (M1) есть один существенный недостаток в сравнении с предыдущими компьютерами Mac с SoC от Intel и портами Thunderbolt 3: они не поддерживают работу с eGPU, т.е. внешними видеокартами, подключаемыми по Thunderbolt 3 (а, если быть вернее, то не поддерживается работы с платами Thunderbolt 3, в которые устанавливаются видеокарты). Так-то можно на Mac с SoC Intel задействовать eGPU и с Thunderbolt 2 и даже Thunderbolt 1, т.к. они, в отличие от USB 3.0/3.1/3.2 Gen 1/2/3 позволяют передавать данные и по линиям PCI-E 3.0, которые и задействуются видеокартами для обмена данными с SoC от Intel, и это можно реализовать, сняв ограничения Apple по части драйверов, что продемонстрировал больше года назад Роман Ястреб в своих видео «MacBook eGPU по Thunderbolt 1/2 - Работает!» и «Обзор Mac Mini 2012 + eGPU Покупаем Mac за 25К!». Ну и тут важное уточнение: пропускной способности Thunderbolt 2 и Thunderbolt 1 не хватит для обеспечения полноценной работы с eGPU в обе стороны. Что я имею в виду под «обе стороны»? Если Вы подключаете внешнюю видеокарту по Thunderbolt к Macbook для того, чтобы повысить производительность графики, отрисованной на встроенном мониторе Вашего Macbook, то данные идут по Thunderbolt в обе стороны: сперва Ваш SoC от Intel передаёт по Thunderbolt 3 внешней видеокарте данные для отрисовки графики, а затем уже отрисованное изображение передаётся обратно в Ваш SoC для вывода через него на встроенный монитор ноутбука. Т.е. это работа в 2 канала. И для неё у Thunderbolt 1 и Thunderbolt 2 уже не хватит пропускной способности и в этой задаче производительность внешней видеокарты будет ограничена пропускной способностью Вашего интерфейса обмена данных с SoC. Но если выводить отрендеренную графику не на встроенный монитор Macbook, а на внешний монитор, то, по большей части, данные от SoC к внешней видеокарте будут передаваться только в одну сторону, т.к. уже отрисованная картина более не возвращается обратно в SoC для вывода на встроенный монитор, а сразу выводится из этой же видеокарты на внешний монитор.

Это важное уточнение, которого не было в видео Романа, поэтому я решил вставить его здесь.

Возвращаясь к Macbook Pro 2020 с SoC Apple M1, у него то USB 4 в полной версии, включающей в себя также и Thunderbolt 3, т.е. поддержка внешних видеокарт должна присутствовать за счёт возможности обмена данными между SoC и видеокартой по линиям PCI-E. Но, тем не менее, внешние видеокарты с ним не работают. Видеокарты определяются системой, но они никак не задействуются при работе. Почему так?

Причина, скорее всего, в драйверах. Если Вы вспомните главу 2.2, то Rosetta 2 не способна транслировать из x86_64 в ARM расширения ядра (kernel extension / kext, аналог драйвера в Windows/Linux). И в принципе это уже даёт ответ на вопрос «а почему внешние видеокарты не поддерживаются». Причина проста: все kext`ы для видеокарт AMD написаны под x86_64 платформу, таких kext`ов под ARM для macOS просто нет, потому ничего и не работает. На переписывание kext`ов под ARM уйдёт столько времени и средств, что вряд ли AMD займётся этим.

Может ли это быть аппаратным ограничением SoC Apple M1? Вряд ли. Нюанс то в том, что USB 4 в своей полной реализации, которая имеет поддержку Thunderbolt 3, должен поддерживать передачу данных по PCI-E.

2.1.1.5 USB4 Host

A USB4 host contains:

- A Host Router.
- A USB 2.0 Host.
- An Enhanced SuperSpeed Host.
- A DisplayPort Source.

A USB4 host can also optionally contain a PCIe controller.

A USB4 host supports 20G USB4 operation (Gen2x2) and optionally 40G USB4 operation (Gen3x2).

A USB4 host is required to support DisplayPort Alt Mode on all of its DFP. See the USB Type-C Specification for a full definition of the requirements and details regarding DisplayPort Alt Mode support on a DFP.

Скриншот 38. Спецификация USB4, файл USB4 1.0 with errata through 20201015 - REDLINE.pdf. Источник: USB-IF.

С полной спецификацией USB4 можете ознакомиться по этой ссылке. Суть в том, что Thunderbolt 3 обеспечивает возможность передачи данных по

PCI-E, и USB4 имеет опциональную поддержку Thunderbolt 3, и компьютеры Apple с SoC Apple M1 имеют именно USB 4 с поддержкой Thunderbolt 3. Т.е. технически чисто на аппаратном уровне должна быть поддержка внешних видеокарт у Macbook Pro, Macbook Air и Mac Mini с SoC Apple M1.



Скриншот 39. Поддержка Thunderbolt / USB 4 у Macbook Pro с SoC Apple M1. Источник: Apple.

Исходя из того, что Apple заявляют о том, что их USB 4 в Macbook Pro имеет поддержку Thunderbolt 3, что требует наличия соответствующего контроллера, на аппаратном уровне поддержка внешних видеокарт должна присутствовать. Я прихожу к выводу, что просто драйверов (или, вернее, kernel extension, kext`ов) нет под Mac на базе Apple Silicon. Появятся ли в будущем драйверы под Mac с архитектурой ARM? Не знаю. Компьютеры Mac с Apple Silicon (M1) были представлены ещё в ноябре 2020. Прошло полгода и пока никаких даже бета версий kext для видеокарт AMD не вышло.

Вы зададитесь вопросом, а как же быть с тем, что USB4/Thunderbolt порты в Mac с M1 урезанные и тянут всего 15 Гбит/сек, что видно на [17:06]?

Нет, на самом деле они не урезанные и тест Романа Ястреба, который состоял в объединении двух компьютеров Mac мостом Thunderbolt показал ровно то, что и обязывает спецификация Thunderbolt 3. Он в M1 не урезан.

Thunderbolt 3 Mode

If a cable and device supporting Thunderbolt are plugged in, the Thunderbolt silicon activates its highest capability mode and configures four high-speed links at either 10 Gbps or 20 Gbps (depending on cable and device support) to support the Thunderbolt transport. This provides bidirectional data rates of 20 or 40 Gbps (Figure 4 illustrates the 40 Gbps case).

Additionally, to fill this Thunderbolt link, the silicon extracts and routes up to 4 lanes of PCI Express Gen 3 (4 x 8 Gbps) and up to two full (4 lane) links of DisplayPort out over the Thunderbolt cable and connector to the device(s) attached downstream from the host system.

The first device in the Thunderbolt link has a few options of how it can use this underlying PCI Express and DisplayPort traffic:

- Consume the data from PCI Express by connecting it to PCI Express enabled devices such as networking, storage, cameras, DSP or FPGA adapters, or perhaps some new custom product
- Consume the DisplayPort links by exposing them to a display panel, or a display connector(s)
- Pass the unused PCI Express or DisplayPort data capabilities down to another Thunderbolt device(s) via a second, daisy chained Thunderbolt 3 port
- Some or all of the above

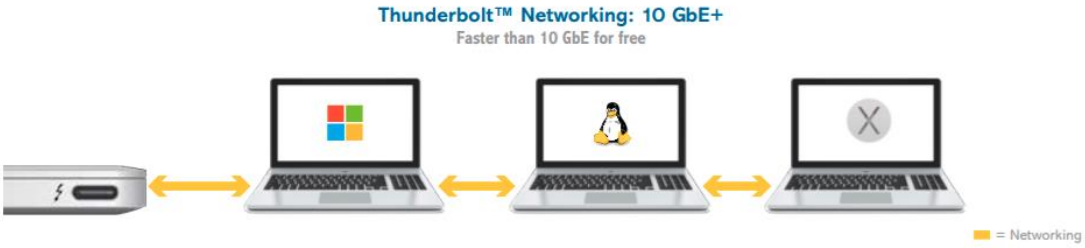
If the Thunderbolt device has two Thunderbolt ports, the dynamic mode configuration described above will be configured on the downstream or second port of the device when another cable and device are plugged in to it. Each mode will still be supported on this downstream connector, and data and display traffic will be provided as demanded to that downstream port.

If each consecutive device is a Thunderbolt device, up to 6 devices in a chain can be supported. At any time if a USB or DisplayPort device is plugged directly into the USB-C connector downstream from any device, then the Thunderbolt daisy chain is terminated at that point.

Thunderbolt Networking Mode

An additional, powerful capability of Thunderbolt also allows the connection of two or more hosts directly (or on separate ends of a Thunderbolt chain) via Thunderbolt. In this case, Thunderbolt has software which creates an IP networking link between these systems (see Figure 5).

This networking mode allows for the Thunderbolt interface to act as a virtual Ethernet adapter and transfer traffic over the PCI Express interface the Thunderbolt silicon has in each host system.



Скриншот 40. Краткая техническая сводка Thunderbolt 3. Информация о Thunderbolt Networking.

Выше я привёл скриншот с 4-ой страницы краткой технической сводки по Thunderbolt 3. Как Вы можете видеть, по Thunderbolt 3 никто и не обещает скорости интернета в 40 Гбит/сек. Более того скажу, что и 40 то Вы нигде не увидите, т.к. 32,4 Гбит/сек – это максимум пропускной способности по 4 линиям PCI-E 3.0 (PCI-E 3.0 x4), ещё 8 Гбит/сек отведены под DisplayPort. Да и из этих 32,4 Гбит/сек Вы получите лишь 25,92 из-за кодирования 8b/10b.

Но главное, что по сетевому мосту по Thunderbolt 3 обещаны лишь 10+ Гбит/сек, т.к. он задействует всего 2 линии PCI-E 3.0, потому в теории макс. пропускная способность - 16,2 Гбит/сек (у Романа на практике – 15.4-15.5).

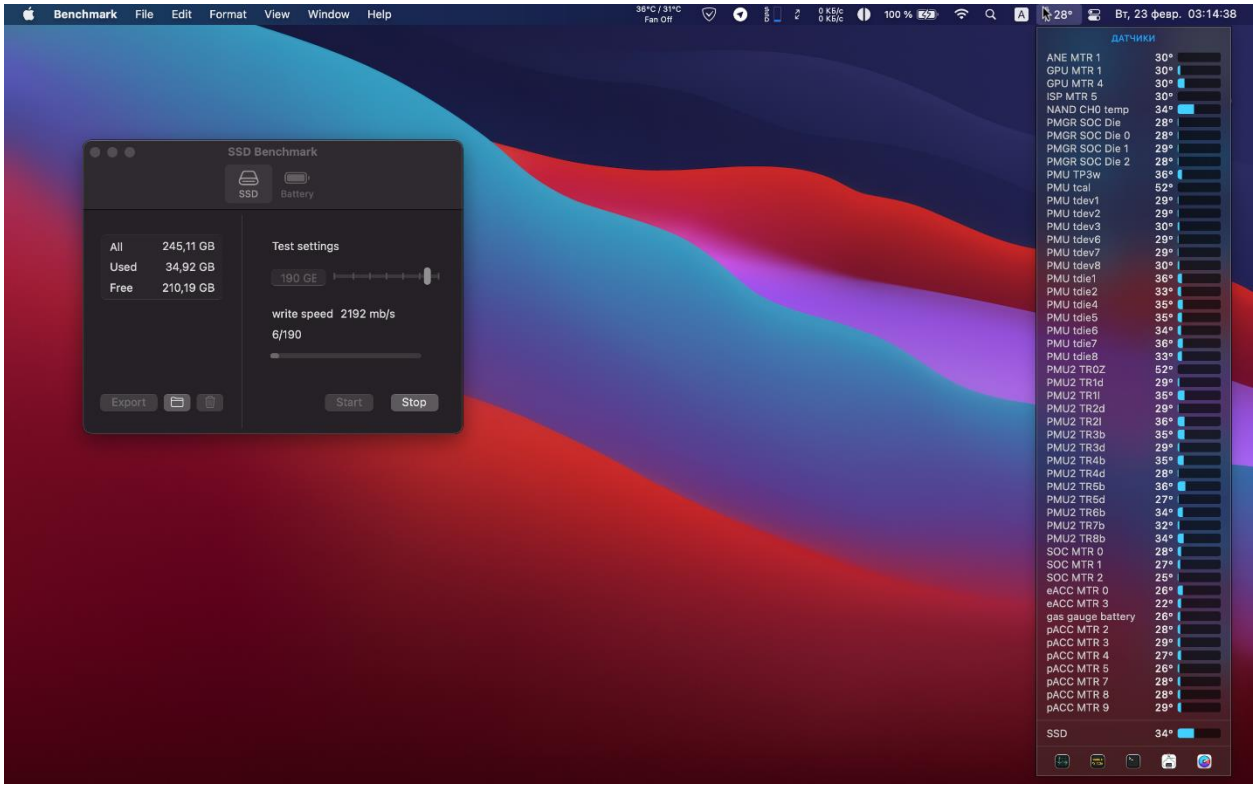
На сайте Intel описано то же самое: обещаны лишь 10 Гбит/сек по сети.

USB4/Thunderbolt 3 в компьютерах Mac с M1 полноценный, он просто и не обязан в режиме моста выдавать больше скорости. Тест не корректен.

3. Тесты SSD накопителя Apple AP0256 и объединённой памяти на подложке с чипом M1.

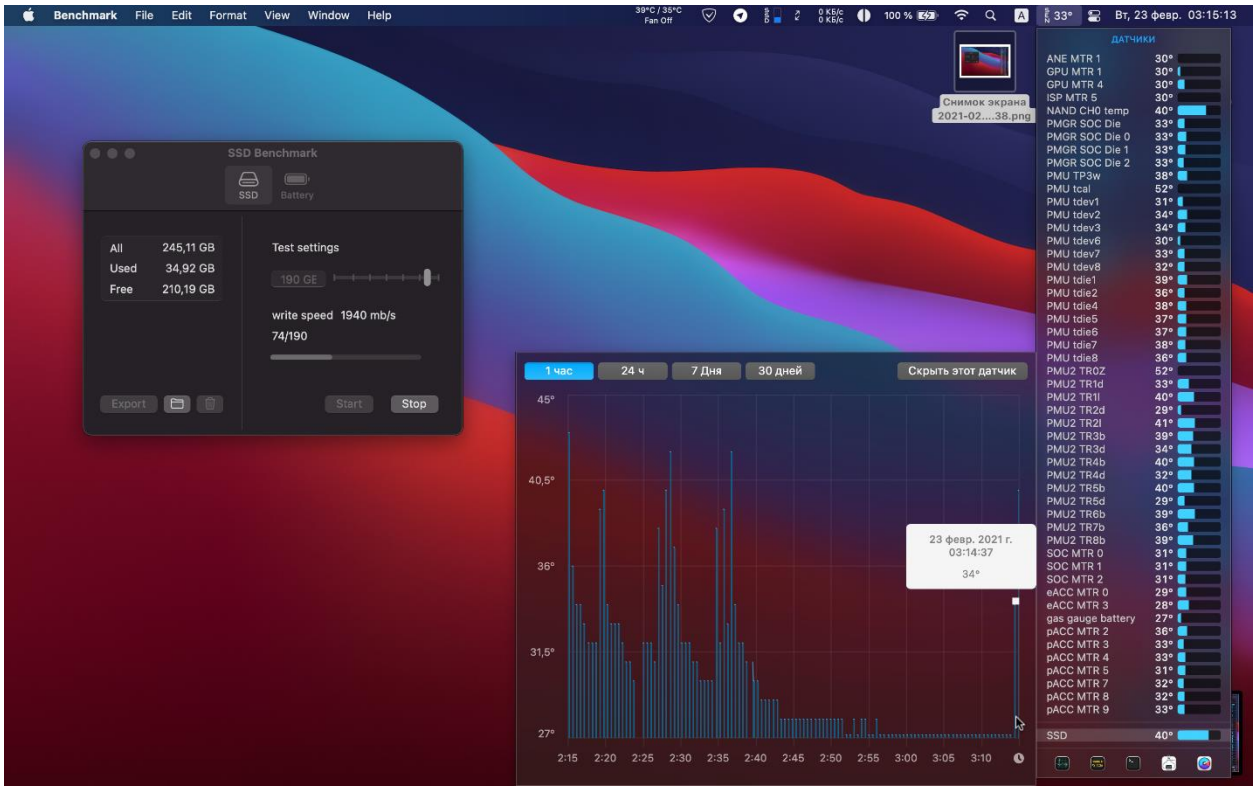
Раз уж я упомянул выше тесты SSD, как-раз с них и начнём. Мы задействуем 3 теста: тест на скорость заполнения SSD от Kavars, CPDT и Blackmagic Disk Speed Test. Мы посмотрим на то, проседает ли скорость записи после заполнения утилитой от Kavars, и оценим, меняется ли скорость записи на SSD в Blackmagic Disk S peed Test и CPDT до и после заполнения.

Спойлер: да, запись проседает, притом сильно. Какого чёрта, Apple?



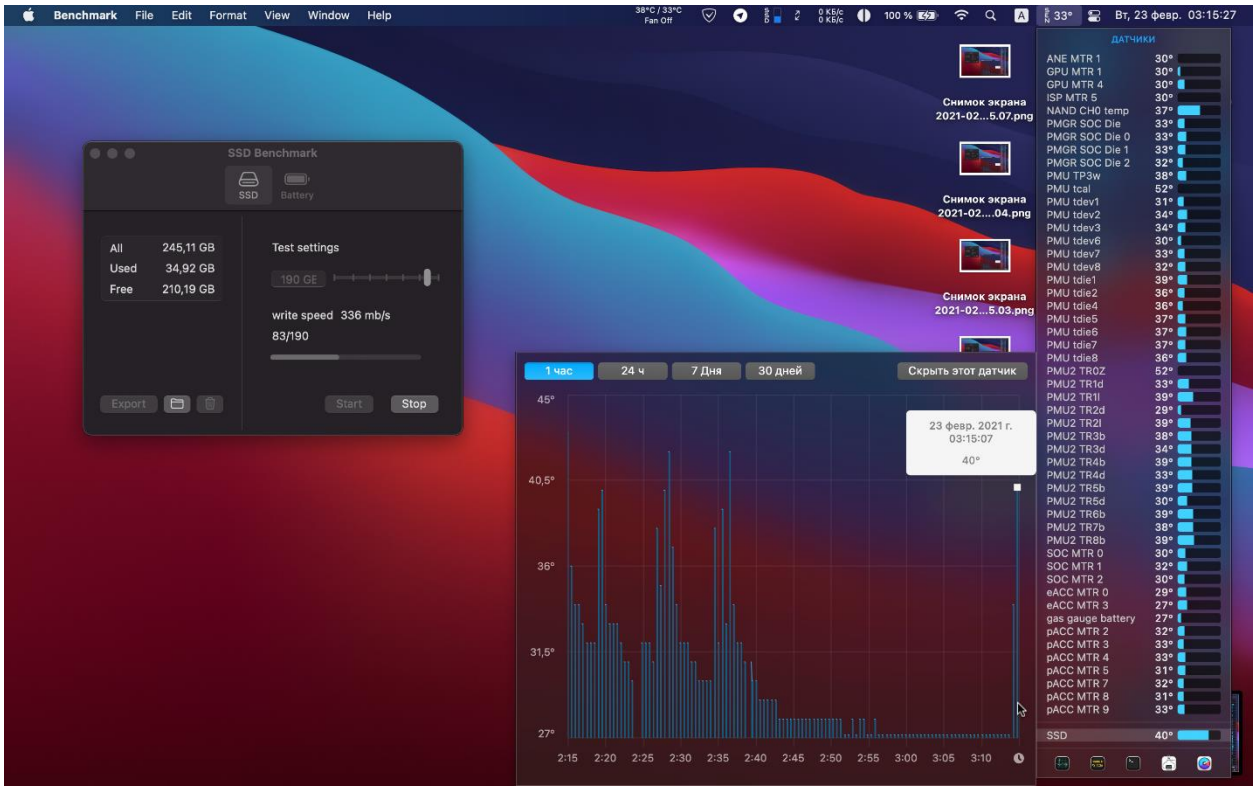
Скриншот 41. Macbook Pro (Apple M1) на 256 GB. Заполнение накопителя утилитой от Kavars. Заполнено 6/190 ГБ.

Первые 9 ГБ запись велась в среднем со скоростью 2.3 ГБ/сек. Тут претензий нет никаких, да и не могут быть претензии, но смотрим далее.



Скриншот 42. Macbook Pro (Apple M1) на 256 GB. Заполнение накопителя утилитой от Kavars. Заполнено 74/190 ГБ.

С 10 ГБ и до 77 ГБ запись велась со средней скоростью 1.94 ГБ/сек. Это хороший показатель, разница в 19% с тем, что было в первые 9 ГБ не особо критична. Это совершенно нормально, что у SSD есть небольшой быстрый кэш.



Скриншот 43. Macbook Pro (Apple M1) на 256 GB. Заполнение накопителя утилитой от Kavars. Заполнено 84/190 ГБ.

На 78-ом ГБ скорость уже составила 1.75 ГБ/сек, на 79-ом – 1.5 ГБ/сек, на 80-ом 1 ГБ/сек, 81-ом – 550 МБ/сек, а после 81 ГБ скорость записи упала до 350 МБ/сек в среднем. Думаете, это временно, и она потом восстановилась до прежних 2 ГБ/сек? Фиг там.



Скриншот 44. Macbook Pro (Apple M1) на 256 GB. Заполнение накопителя утилитой от Kavars. Заполнено 189/190 ГБ.

Короче говоря, после заполнения SSD накопителя на 77 ГБ или 45,7% ((35+77)/245*100% = 45,7%), его скорость стремительно падает и уже через менее чем ещё 5 ГБ скорость падает до 352 МБ/сек в среднем. В сравнении с 1.94 ГБ/сек на протяжении записанных 10-77 ГБ падение составило аж 5,5 раз. Сравнивать с изначальными 9 ГБ смысла особо нет, т.к., скорее всего, высокая скорость при записи первых 9 ГБ обусловлена SLC-кэшем.

У него скорость проседает как у одного из худших SSD в ноутбуках, что мы тестировали. Однако, если смотреть на среднее значение за всё время тестирования, выходит, что в среднем скорость составляла 1031 МБ/сек. На запись 190 ГБ данных были затрачены 352,9 секунды.

Ещё один важный нюанс: macOS, в отличие от Windows, считает пространство в гигабайтах (GB), а не гибибайтах (GiB). Разница в том, что гигабайты – это 10^9, а гибибайты – 2^30. Т.е. гигабайт – это 1.000.000.000, а гибибайт – 1.073.741.824. Также и мегабайт – это 10^6 или 1.000.000, а

мебибайт – это 1.048.576. Т.е. чтобы сравнивать показатели с теста SSD в утилите benchmark от Kavars, нужно перемножать все мегабайты на 1.000.000 и делить на 1.048.576, а все гигабайты перемножать на 1.000.000.000 и делить на 1.073.741.824. Собственно, именно из-за различия в том, как считают пространство эти Операционные Системы, и выходит, что покупая SSD на 1 ТБ и подключая его к компьютеру под управлением macOS Вы видите доступный 1 ТБ пространства, а подключая его к Windows Вы видите 1.000.000.000.000/1.073.741.824 = 931,3 GiB (Гибибайт). Аналогично вместо 256 GB в винде на накопителе Вы увидите 238,4 GiB. Если не верите мне, просто прочитайте про это статью на Википедии.

Измерения в байтах								
ГОСТ 8.417—2002			Приставки СИ		Приставки МЭК			
Название	Обозначение	Степень	Название	Степень	Название	Символ		Степень
байт	Б	10^0	—	10^0	байт	B	Б	2^0
килобайт	Кбайт	10^3	кило-	10^3	кибибайт	KiB	КиБ	2^{10}
мегабайт	Мбайт	10^6	мега-	10^6	мебибайт	MiB	МиБ	2^{20}
гигабайт	Гбайт	10^9	гига-	10^9	гибибайт	GiB	ГиБ	2^{30}
терабайт	Тбайт	10^{12}	тера-	10^{12}	тебибайт	TiB	ТиБ	2^{40}
петабайт	Пбайт	10^{15}	пета-	10^{15}	пебибайт	PiB	ПиБ	2^{50}
эксабайт	Эбайт	10^{18}	экса-	10^{18}	эксбибайт	EiB	ЭиБ	2^{60}
зеттабайт	Збайт	10^{21}	зетта-	10^{21}	зебибайт	ZiB	ЗиБ	2^{70}
йоттабайт	Ибайт	10^{24}	йотта-	10^{24}	йобибайт	YiB	ЙиБ	2^{80}

Скриншот 45. Различие между мегабайтами и мебибайтами, гигабайтами и гибибайтами. Информация из Википедии.

И так выходит, что 1031 МБ/сек в системе счисления, которую использует Windows, это 983 MiB/sec. Это, с одной стороны, великолепный результат, который уступает лишь RAID-массиву из 2-ух SSD Samsung PM981a в ноутбуке Asus ROG Zephyrus DUO (GX550LXS). Однако, с другой стороны, это чёртово падение в 5 раз и после записи 82 ГБ/сек мы по факту наблюдали скорость в 352 МБ/сек в среднем (или 335,5 MiB/sec, если переводить в мебибайты винды). Это же жуть как мало для PCI-E SSD и, в отличие от многих ноутбуков на Windows, в случае с Macbook Pro Вы не можете просто взять и заменить SSD, т.к. он распаян на материнской плате.

Тем, кто не понимает, что тут вообще происходит, и от чего зависит это само падение или же постоянство скорости записи, рекомендую

ознакомиться с видео «SSD. Как работают, почему ломаются? SLC, MLC, TLC, QLC» на канале «Этот Компьютер». Там автор канала наглядно пояснил все различия между SLC, MLC, TLC и QLC накопителями, и как отличить плохую память от хорошей. Apple не называет типа памяти, используемого в SSD в их Macbook, поэтому тут мы можем опираться лишь на наши результаты тестирования.

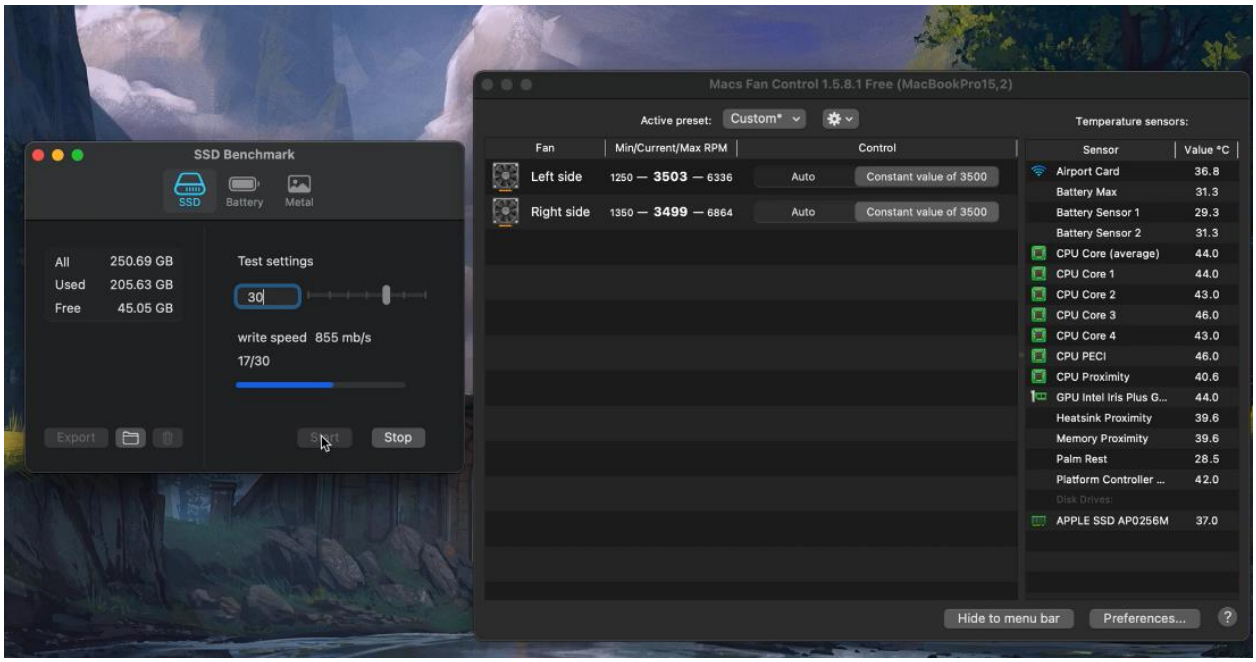
Да, Apple задействует уплотнение уже записанных данных, за счёт чего после заполнения накопителя на 45% через некоторое время следующие ещё 40 ГБ будут записываться на той же скорости в 1.9 ГБ/сек, но объём пространства, доступного для заполнения в однобитном режиме (короче говоря, в режиме максимальной скорости, для подробностей см. видео от «Этот Компьютер») с каждым последующим записанным гигабайтом будет сокращаться, и Вы рано или поздно, если будете записывать на SSD большие объёмы данных, столкнётесь с падением до этих пресловутых 350 МБ/сек.

Я бы и слова против не сказал, если бы SSD можно было заменить в Macbook. Не устраивает? Не вопрос, покупай тот SSD, который тебя устроит, и используй его. Но такой опции нет и никогда не было в Macbook с USB Type-C или Thunderbolt 3, т.е. начиная с 2016 года в линейке Macbook Pro заменить SSD нельзя, и в Macbook Pro с M1 тенденция осталась та же, и заменить SSD нельзя.

Есть ли плюсы у этого SSD? Да, температура.

Обратите внимание на данные с датчиков в iStat Menus: температура SSD за всё время тестирования заполнением не превысила 42 градусов. Чтобы Вы понимали, при сопоставимой средней скорости записи мой SSD Samsung 970 Evo Plus на 1 TB при заполнении греется до 116 градусов под нагрузкой, а SSD SK Hynix на 256 GB HFS256GD9TNG-62A0A, который изначально поставлялся с моим ноутбуком Xiaomi, греется при работе до 90 градусов, при этом уже после 5-10 ГБ сбавляя скорость заполнения до тех же 300-350 MiB/sec, что и у Macbook Pro 13 2020 с чипом M1. Чувствуете разницу? Скорость у Mac та же, но температура не 90 градусов, а 42 в пике.

Мы на днях тестировали ещё Asus ROG Strix G15 с SSD SK Hynix на 1 TB и у него скорость составляла в среднем 980-985 MiB/sec, т.е. примерно как у Macbook Pro 13 с M1, но грелся он до 82 градусов в пике.



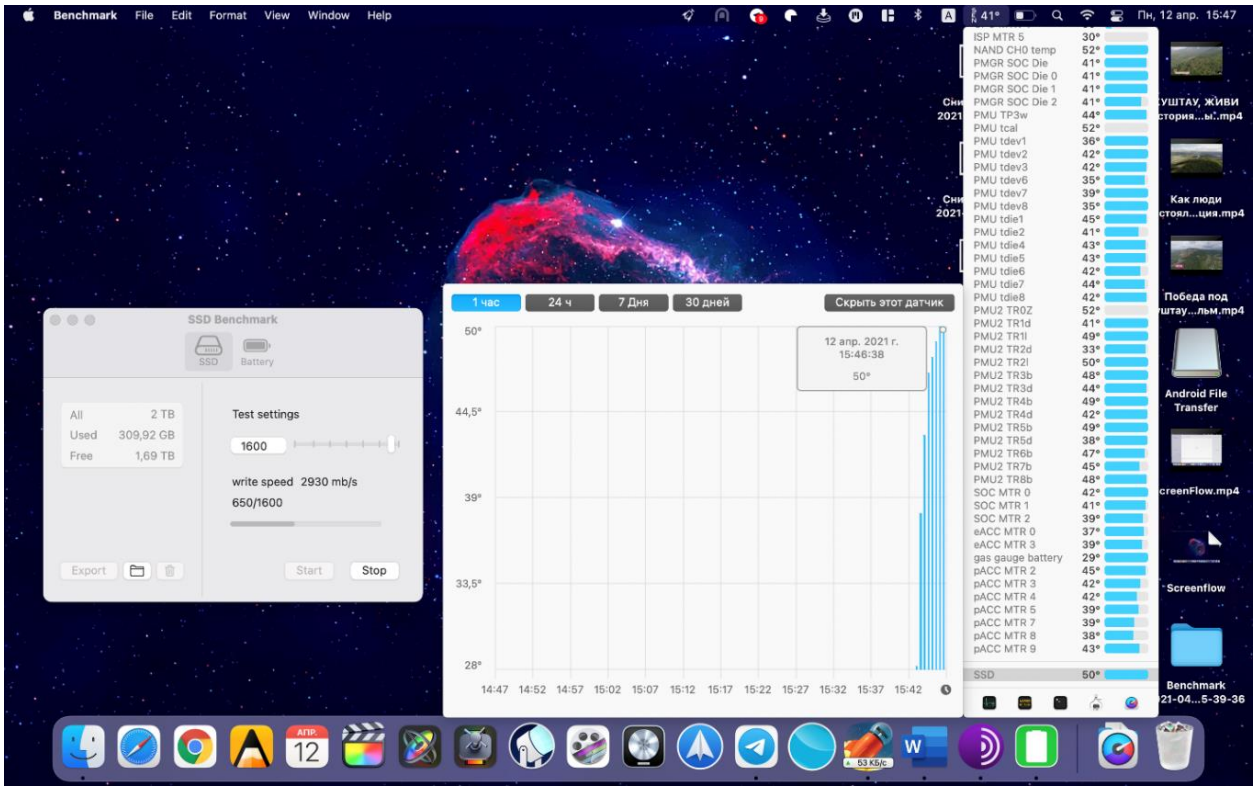
Видео 2. Macbook Pro 13 2018 2x TB3 (i5 8259U). Тест SSD утилитой Kavars.

Впрочем, Apple не первый год в свои Macbook устанавливают очень холодные SSD. С MBP 13 2018 моего хорошего друга, Кирилла Варшамова, та же история: он греется не более чем до 40 градусов. Конечно, хотелось бы протестировать его на более длительной записи, но у Кирилла это рабочий ноутбук, и он не может его просто в момент очистить от всех своих данных.

В общем, палка о двух концах. С одной стороны, у Вас скорость SSD значительно снизится после заполнения, а с другой – он не греется, т.е., за сохранность данных можно не переживать. Я бы предпочёл заменить SSD, чтобы скорость была постоянно не менее 1 GiB/sec, но в Macbook Pro он уже года с 2016 не съёмный, поэтому выбора уже после покупки у Вас не будет (разве что продать Macbook и купить другой ноутбук или купить тот же Macbook в другой комплектации). Про замену компонентов можно забыть.

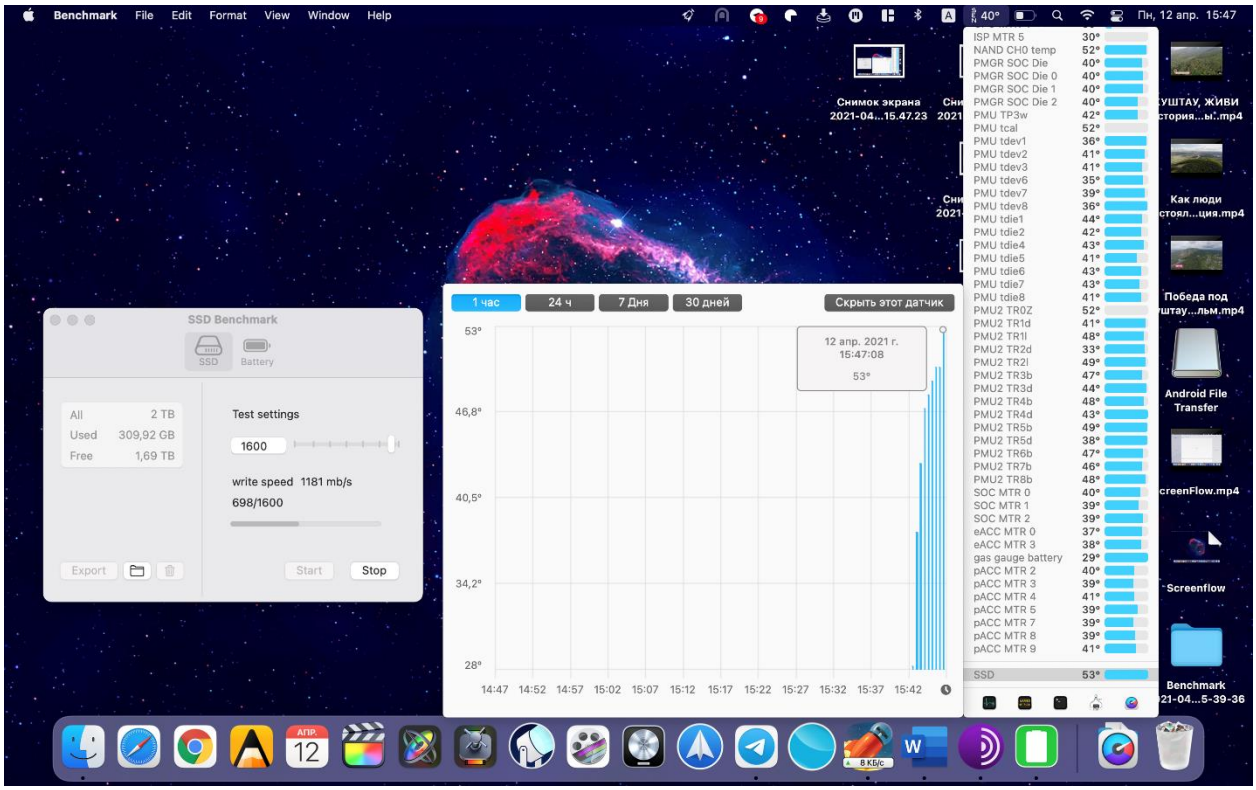
Я в главе 2.7 упоминал, что нам удалось запустить утилиту Benchmark от Kavars на экземпляре Macbook Pro с Apple M1 на 2 TB Стаса, пересобрав её из исходного кода через XCode под «My Mac». Что из этого вышло в итоге? Лучше или хуже там память, чем в 256 GB модели?

Спойлер: лучше, причём намного.



Скриншот 46. Macbook Pro (Apple M1) на 2 TB. Заполнение SSD накопителя утилитой от Kavars. 650/1600 ГБ.

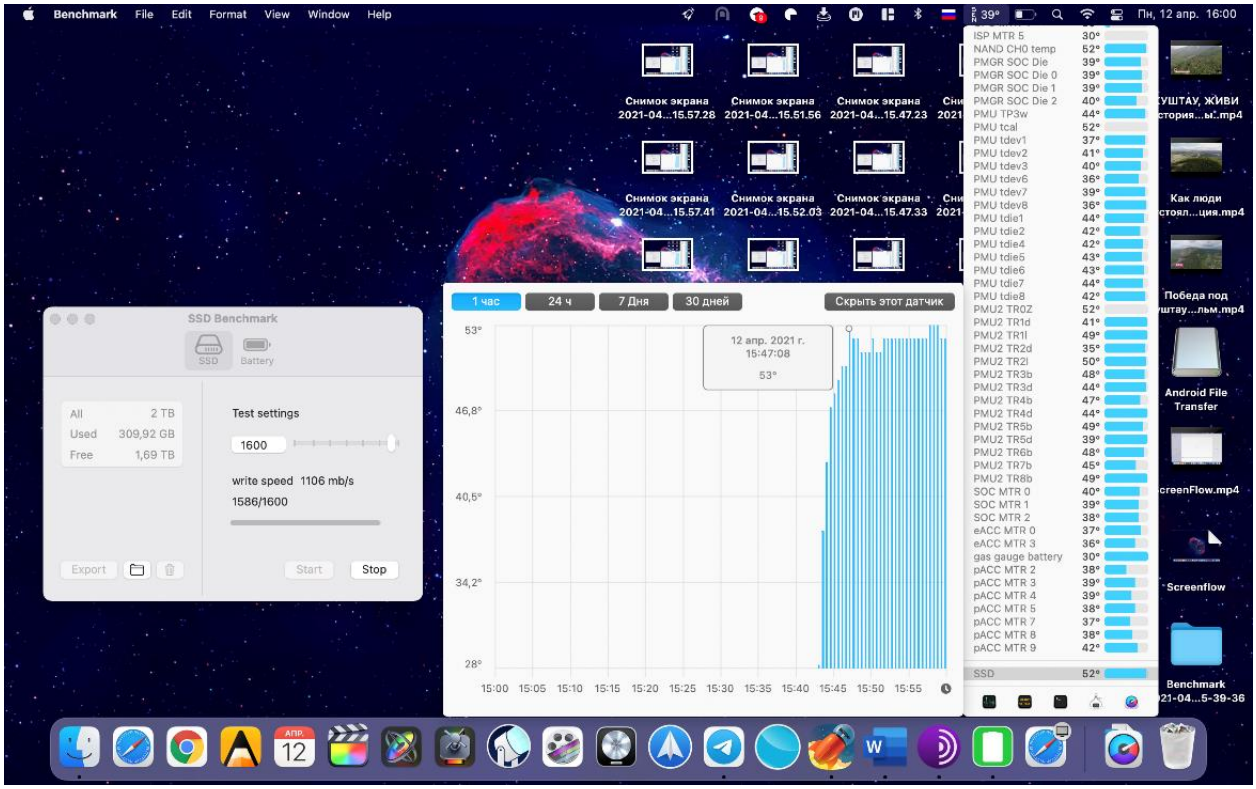
С экземпляром Стаса на 2 TB ситуация с самого начала обстоит иначе: первый 1 GB был записан за 0.2748 секунды, т.е. скорость записи в самом начале составила 3638 MB/sec или 3470 MiB/sec, а далее со 2-го по 682-ой GB запись велась со средней скоростью 3027 MB/sec или 2887 MiB/sec.



Скриншот 47. Macbook Pro (Apple M1) на 2 TB. Заполнение SSD накопителя утилитой от Kavars. 698/1600 ГБ.

Накопитель Стаса до теста был заполнен на 310 GB. Т.е. пока накопитель не был заполнен на 992 GB, или на 49,6%, или же пока мы не заполнили его ещё на 34,1% (с 15,5% до 49,6%) его скорость составляла 3 GB/sec стабильно. 256 GB модель более чем в 1.5 раза медленнее здесь же.

683-ий GB был записан за 2,5 секунды, т.е. скорость при его записи составила всего 398 MB/sec или 380 MiB/sec. Однако, последующая запись с 684-го ГБ по 1600 ГБ велась со скоростью 1180 MB/sec или 1125 MiB/sec.



Скриншот 48. Macbook Pro (Apple M1) на 2 TB. Заполнение SSD накопителя утилитой от Kavars. 1586/1600 ГБ.

Если похвалить SSD в 256 GB модели я не могу, то в модели на 2 TB он один из лучших, что мы тестировали в ноутбуках. Температура не превышала 53 °C, тогда как у других ноутбуков она при записи составляла 80+ °C. Ещё и средняя скорость записи при заполнении составила 1967 MB/sec или 1876 MiB/sec. Да, после заполнения на 49,6% средняя скорость записи снижается с 3027 MB/sec (2887 MiB/sec) до 1180 MB/sec (1125 MiB/sec), но даже так она самая быстрая среди всех ноутбуков, что мы тестировали ранее. Предыдущий рекорд принадлежал Asus ROG Zephyrus DUO с 2x 1 TB SSD Samsung PM981a в RAID 0 (1004,4 MiB/sec), а с одним SSD (без RAID) рекорд принадлежал Asus ROG Strix G15 с 1 TB SSD SK Hynix HFM001TD3JX013N (980,15 MiB/sec). Macbook с M1 на 2 TB удивляет.

Вернёмся к моему экземпляру Macbook Pro с Apple M1 на 256 GB.



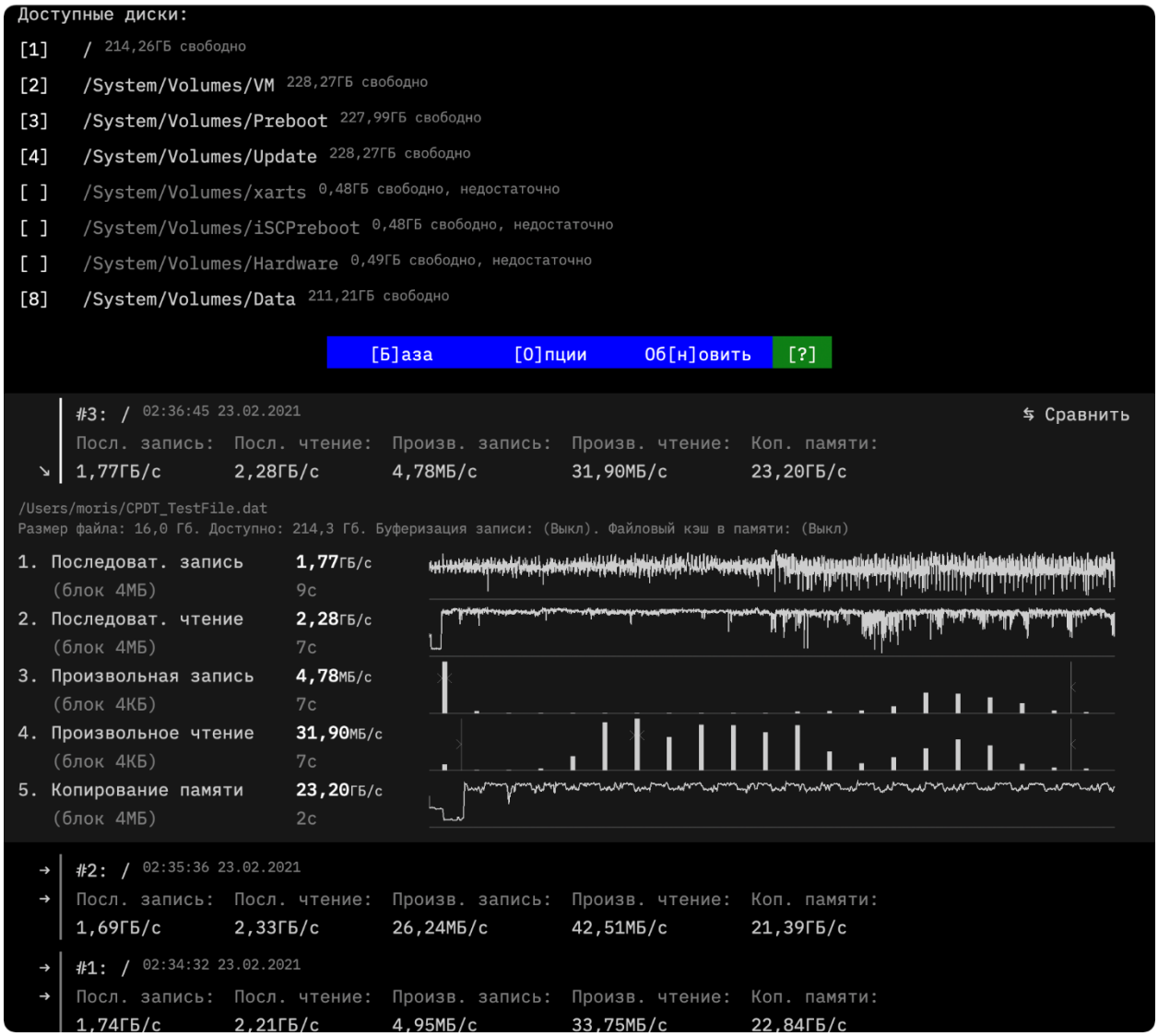
Скриншот 49. Macbook Pro 13 2020 (Apple M1). Blackmagic Disk Speed Test до заполнения накопителя.

В Blackmagic Disk Speed Test до заполнения в целом скорость записи колебалась до заполнения от 2 ГБ/сек до 2.2 ГБ/сек с разовыми просадками.



Скриншот 50. Macbook Pro 13 2020 (Apple M1). Blackmagic Disk Speed Test после заполнения накопителя.

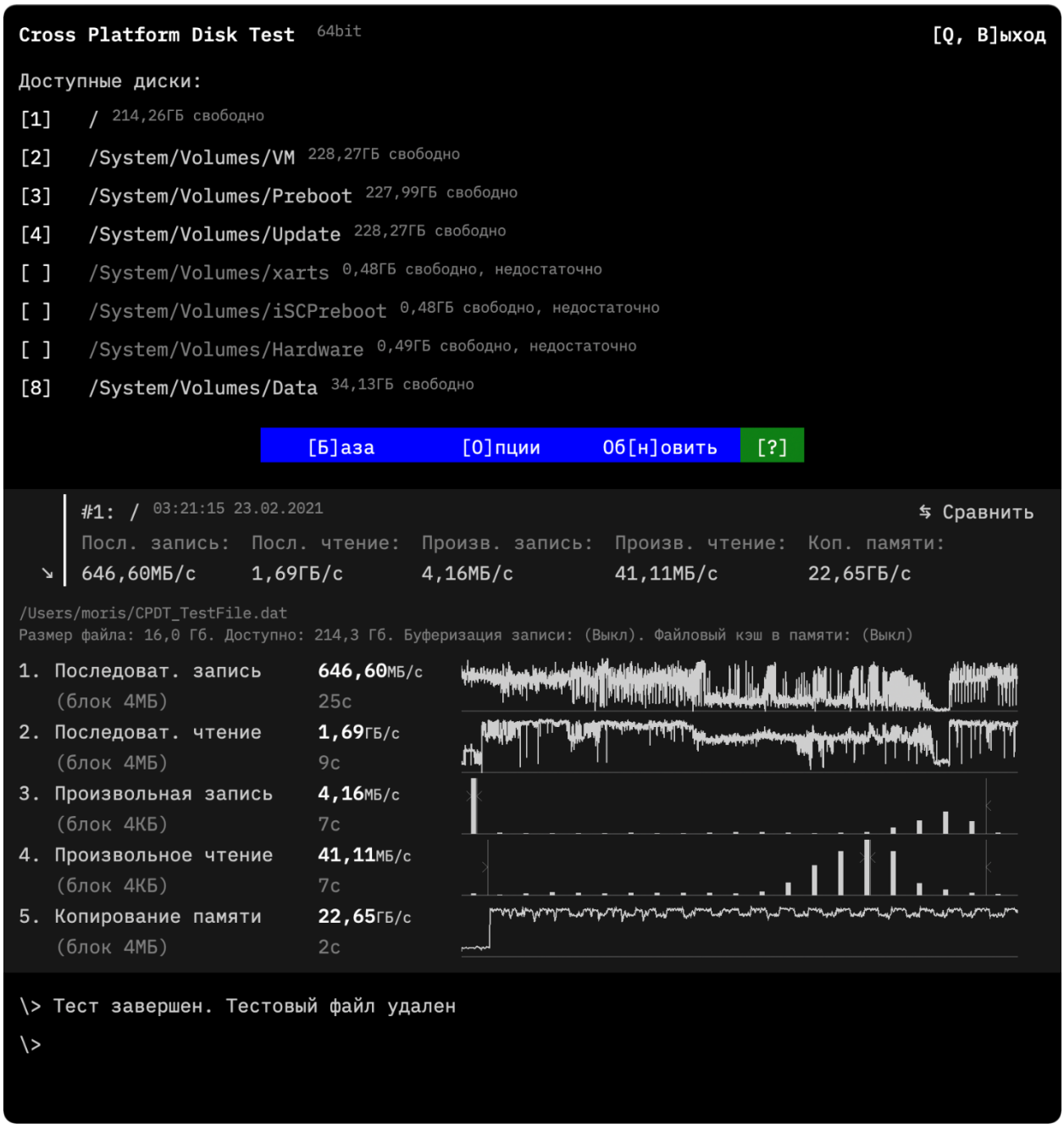
После заполнения тут ничего особо не поменялось.



Скриншот 51. Macbook Pro 13 2020 (Apple M1). CPDT. Графики. Результат до заполнения накопителя.

Выше Вы видите результаты теста в CPDT до заполнения накопителя, которые довольно-таки высоки. Обратите внимание на то, что график последовательной записи намного более ровный до середины, т.е. когда заполняются первые 8 ГБ, которые у нас в SSD отведены под кэш, а после график становится значительно более дёрганным. В этом нет ничего криминального, это просто демонстрация того, как ведёт себя SSD до исчерпания кэша и после даже тогда, когда он ещё не заполнен.

Но что ужасно удручает: чрезвычайно низкая скорость произвольной записи SSD в 1 поток. Настолько низкой скорости случайных операций у нас не было ни на одном протестированном ранее ноутбуке. Тут либо CPDT сбоит, либо у SSD в Macbook с M1 действительно имеются проблемы с произвольной записью, во всяком случае, в однопоточном режиме.



Скриншот 52. Macbook Pro 13 2020 (Apple M1). CPDT. Графики. Результат после заполнения накопителя.

После заполнения у нас весь график последовательной записи стал рваным, проявились глубокие просадки с середины графика, и они слишком отчётливы, чтобы их просто пропустить. Скорость записи = печаль. Что странно и чего ранее мы не наблюдали в других ноутбуках: рваным стал даже график последовательного чтения, хотя обычно заполнение SSD накопителя не должно влиять на скорость чтения. С произвольной/случайной записью ситуация не поменялась, а вот произвольное/случайное чтение стало чуть лучше, но это можно списать на погрешность измерения.

Если верить CPDT, скорость копирования данных в объединённой памяти тут такая же, как и у оперативной памяти LPDDR4X в ноутбуке Asus Zenbook Flip S (UX371E), работающей на частоте 2133 МГц (LPDDR4X-4266). С одной стороны, это великолепный показатель, а с другой не совсем понятно, а почему Apple не установили сюда LPDDR5. Во всяком случае, для Вас не будет новостью, если Вы читали текст с обзором Zenbook Flip S, что задержки у памяти LPDDR4X куда выше, чем у классической оперативной памяти DDR4. Однако, благодаря тому что объединённая память у Mac с M1 размещена вплотную к чипу M1, и маршрут, по которому данные должны пройти от процессора к памяти и обратно, минимизирован до минимально возможного для значения, вполне возможно, что Apple сумели достичь высокой производительности, низких задержек и низкого энергопотребления.

Но и тут есть 1 нюанс: CPDT работает через Rosetta 2. Нативной версии под Mac на Apple Silicon ещё не публиковали. Вполне возможно, что из-за того, что тут происходит предварительная трансляция (заблаговременный перевод) уже готового машинного кода под x86_64 в машинный код для ARM, мы наблюдаем куда меньшую эффективность взаимодействия процессора с объединённой памятью и значения тут должны быть несколько Выше. Вот только удостовериться мы в этом никак не можем. Единственные 2 теста под macOS, которые хоть как-то более-менее явно отражают скорость взаимодействия процессора с объединённой памятью это CPDT и NovaBench... И оба пока существуют только в версии для Intel... Никакой AIDA64 под macOS нету. Работаем с тем, что есть. Простите меня, дорогие маководы, но я не могу параллельно с тестированием ноутбука ещё разрабатывать под него софт для тестирования (тем более на языках, с которыми я не знаком), так что будем довольствоваться теми тестами, которые мы можем провести. Ещё раз спасибо Кириллу (Kavars) за то, что написал утилиту для теста SSD на стабильность скорости при заполнении. Без него тут было бы сильно меньше информации.



Скриншот 53. Информация о приложениях CPDT и NovaBench. Пока что работают только через Rosetta 2.

Раз уж зашла речь про NovaBench, проверим заодно и его, ведь он включает в себя бенчмарки CPU, GPU, объединённой памяти и SSD.



Скриншот 54 Результаты тестов ноутбуков в Novabench: Macbook Pro M1, Xiaomi, Macbook Pro i5 1038NG7.

Сравним данные с Macbook Pro с M1 с моим Xiaomi и MBP 13 2020 моего товарища, Владимира Баруса, которого я упоминал в начале статьи. Как видите, по результатам теста NovaBench, скорость оперативной памяти в МБ/сек в ноутбуке Macbook Pro 13 2020 с чипом Apple M1 на 28,5% превышает таковую у оперативной памяти в ноутбуке Macbook Pro 13 2020 с 4 TB3 и чипом Intel Core i5 1038NG7, и на 65,5% превышает таковую у оперативной памяти моего Xiaomi. Напомню, что эти значения могут на деле быть несколько выше и ограничением тут может выступать не столь высокая скорость операций из-за Rosetta 2. Впрочем, даже так видно, что объединённая память у нового Macbook значительно быстрее.

Почему я всё время зову эту память объединённой, а не оперативной? Всё дело в том, что у CPU, GPU, NPU и ISP в SoC Apple M1 используют один единый пул памяти, т.е. память не разграничивается на оперативную память и видеопамять, и нет тут ограничения по занимаемому объёму памяти для каждого отдельного компонента в SoC. Если у Вас приложение требует 12 ГБ видеопамяти, никаких проблем не будет, пока Вы остальные 4 ГБ не забьёте (а забиваются они быстро, т.к. система хранит в памяти много кэша).



Скриншот 55. Xiaomi Mi Notebook Pro GTX. Об этом Mac.

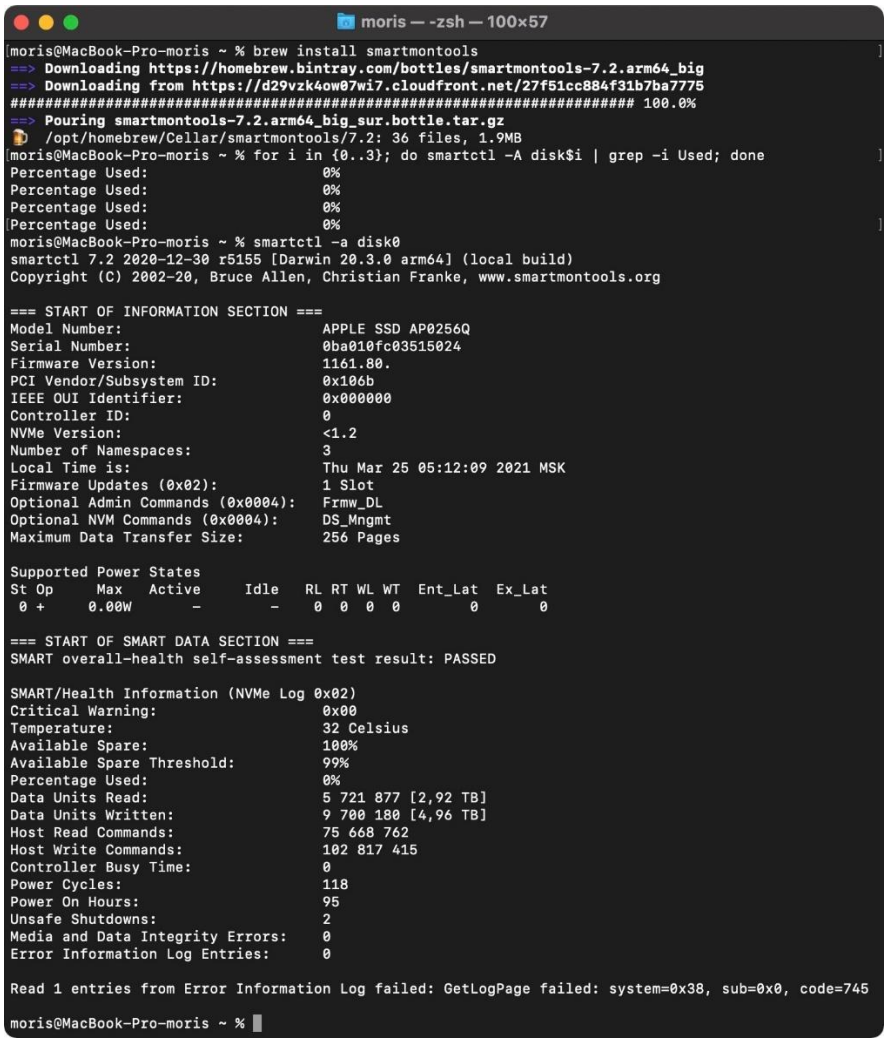
Привожу изображение с начала статьи, чтобы Вы обратили внимание. У графики Intel имеется лимит по количеству видеопамяти, которая она может зарезервировать под свои нужды среди оперативной памяти. И система Вам об этом лимите сообщает в панели «Об этом Mac». У Macbook Pro с чипом M1 такого ограничения нет, потому это и зовётся объединённой памятью, а не оперативной памятью, ведь у Вас оперативная память в полной мере является и видеопамятью. Было бы лучше, будь тут в таком случае, хотя бы, 20 GB объединённой памяти, ведь именно 16 ГБ оперативной памяти и 4 ГБ видеопамяти требовалось ранее Macbook Pro 16” для решения тяжёлых задач. Впрочем, думаю, что с релизом следующих Macbook Pro на чипе Apple M1X (или Apple M2) будет возможность выбора моделей с 32 GB памяти.

Что ещё нам даёт понять Novabench? Что скорость чтения SSD накопителя практически упирается в потолок пропускной способности PCI-E 3.0 x4. Мы подобного не наблюдали ранее в тестах Blackmagic Disk Speed Test и CPDT, но Novabench нам это наглядно проиллюстрировал, отобразив скорость чтения в 3.2 ГБ/сек. По производительности Macbook Pro с чипом M1 обходит оба представленных ноутбука как по CPU, так и по GPU. Он обходит их при том, что ранее в его же корпусе с 2 портами Thunderbolt 3 поставлялся Macbook Pro 13 с чипом i5 8257U или i7 8557U, оба при этом имеют с завода расчётный TDP в 15 Ватт (столько для них указывает Intel). В ноутбуке Xiaomi у меня TDP повышен изготовителем ноутбука до 25 Ватт, а какой TDP по факту у Macbook Pro 13 с чипом M1, и какой TDP по факту у Macbook Pro 13 2018 с 4 TB3 (i5 8259U) и Macbook Pro 13 2020 с 4 TB3 (i5 1038NG7) Вы узнаете уже в следующем главе. Обещаю, что будет интересно.

По графике: одним из факторов, который позволил в Novabench новому Macbook добиться отрыва аж в 2 раза от своего предшественника классом выше как-раз и является объединённая память, которая теперь в добавок ещё и размещена прямо у SoC Apple Silicon. На мой Xiaomi по графике не смотрите ведь, как я уже упоминал, чипы, которые Intel эксклюзивно первыми поставляли Apple, имеют повышенную графическую производительность за счёт большего числа вычислительных блоков у интегрированной графики. А почему же такой отрыв от Macbook Pro 13 2020 с 4 TB3 на базе чипа Intel Core i5 1038NG7: нет ограничений на объём видеопамяти, минимизированы задержки за счёт более тесного контакта объединённой памяти с SoC с CPU+GPU, и много наработок, на результаты которых нам ещё предстоит посмотреть в этой статье.

Но перед этим мы начнём следующую главу с того, что меня огорчило, а именно - троттлинг-теста. Сейчас мы зафиксируем в сравнительной таблице данные из CPDT и скорости заполнения накопителя. NovaBench и Blackmagic Disk Speed Test фиксировать в таблице не будем, т.к. этот софт доступен только под macOS и потому сравнивать показатели его в таблице с Windows аппаратами мы не сможем.

Прежде, чем мы завершим главу с SSD, нужно задаться ещё одним вопросом, который фигурировал в новостях изданий 3DNews, iXBT, cnews, mobile-review, Rozetked, 9to5Mac, MacRumors и т.д.: что там с износом SSD?



Скриншот 56. Macbook Pro 13 2020 (Apple M1). Информация об SSD (S.M.A.R.T..) из утилиты smartctl.

Да нормально всё. Я пишу эту часть текста спустя 2 месяца использования и после записи 5 ТБ данных. Я тестировал на Mac SSD несколько раз (надо же удостовериться в повторяемости результатов), я загружал и удалял игры несколько раз, я проверял много софта (для чего его устанавливал и удалял), много раз записывал видео с экрана и, в общем-то, износ не составил даже 1%. Возможно, если активно эксплуатировать модель с 8 ГБ объединённой памяти, ситуация будет значительно хуже, т.к. при ресурсоёмкой работе для компенсации недостающей памяти DRAM на ней будет чаще задействоваться swap (файл подкачки) на SSD. Вот только кто будет в здравом уме для ресурсоёмкой работы брать 8 ГБ модель? Раздули из мухи слона, в общем. Не стоит по этому поводу заморачиваться, всё ок.

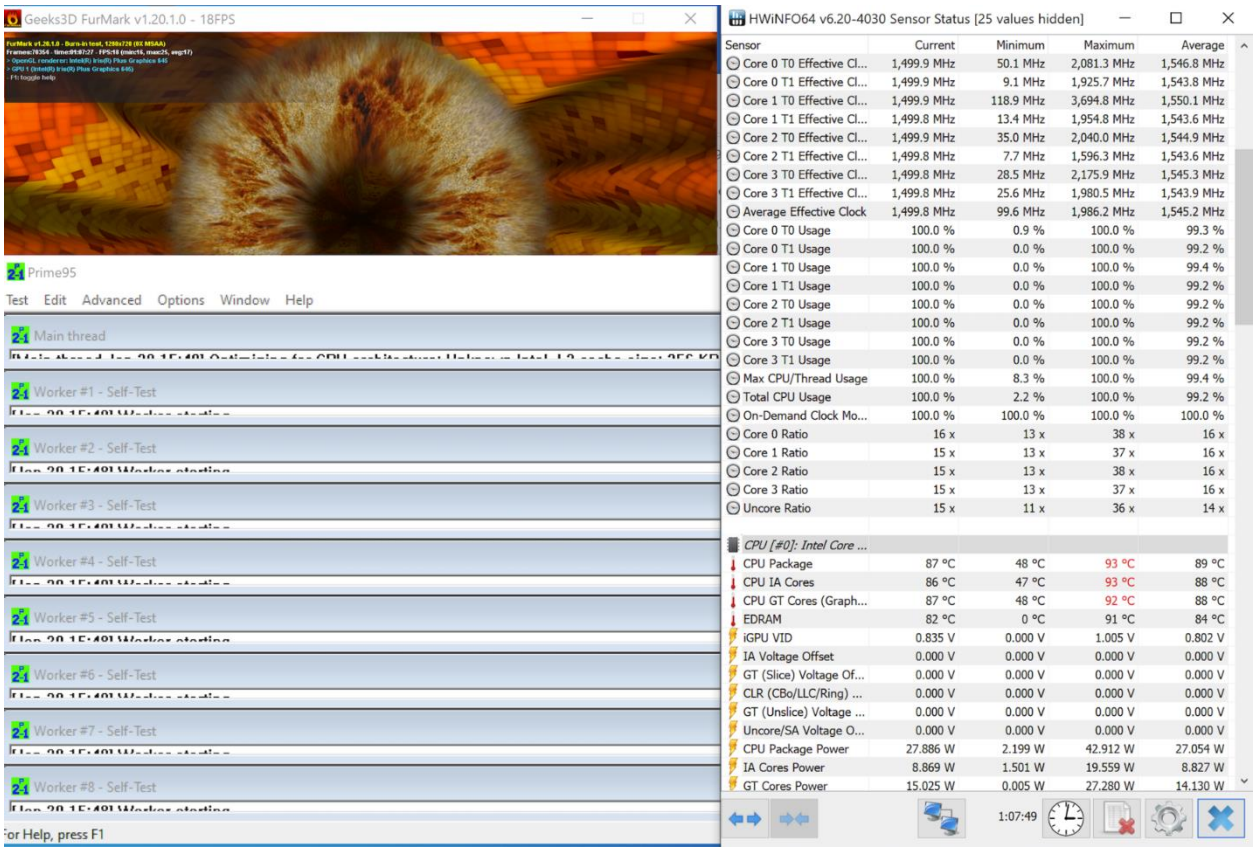
	Acer Nitro 5 AN515-44 (WDC SN530 512GB)	Asus ROG Zephyrus DUO GX550LXS (RAID 0 - 2x1TB Samsung PM981a)	RedmiBook 16 2020 (Samsung PM881 512GB)	Dell XPS 17 9700-6727 (Toshiba XG6 - KIOXIA KXG60ZNV1T02)	Asus ROG Strix Scar 17 G732LXS (RAID 0 - 2x1TB Intel 660p)	Asus ZenBook Flip S UX371E (1 TB WDC PC SN730)	HP Omen 15 15-en0037ur (512 GB Samsung PM981a)	Macbook Pro 13 2020 Z11B0004T Apple M1 (256 GB Apple AP0256)
Powershell - Avg Allocation Speed (MiB/s)	437,68	1004,40	319,34	838,65	119,58	718,62	753,71	
Powershell - TRIM speed (GiB/s)	60,51	150,50	82,25	80,23	17,39	56,75	55,57	
Swift + Obj-C allocation - max temperature								42
Swift + Obj-C - Avg Allocation Speed (MiB/sec)								983,08
CPDT - seq. read (MB/s) - before	1950	2340	480,31	2170	2420	2360	2620	2280
CPDT - seq. read (MB/s) - after	1940	2620	478,72	2120	2390	2370	2720	1690
CPDT - seq. write (MB/s) - before	504,11	1340	350,42	1410	1390	1450	1210	1770
CPDT - seq. write (MB/s) - after	487,42	1490	337,30	1230	568	1420	954	647
CPDT - rand. read (MB/s) - before	37,33	47,15	26,90	38,71	46,99	33,56	46,62	31,90
CPDT - rand. read (MB/s) - after	37,24	35,72	22,41	30,46	46,27	32,78	38,75	41,11
CPDT - rand. write (MB/s) - before	68,80	71,44	38,54	72,04	82,55	57,81	59,76	4,78
CPDT - rand. write (MB/s) - after	67,31	76,77	37,60	72,56	77,59	58,58	59,68	4,16
CPDT - Memory Copy (GB/s)	6,06	9,32	5,13	11,53	13,58	23,24	6,41	23,20

Таблица 1. Сравнительная таблица с результатами тестирования SSD в ноутбуках.

Выводы по SSD: SSD в 256 GB модели слабоват. У модели с 256 GB SSD скорость записи падает в 5.5 раз с 1.9 ГБ/сек до 350 МБ/сек после заполнения на 45%. 2 ТБ SSD тоже проседает после заполнения на 49.6%, но всего в 2.57 раза: с 3 ГБ/сек до 1.2 ГБ/сек в среднем. Средняя скорость в течение всего заполнения у него тоже выше – 1876 MiB/sec вместо 983 MiB/sec. За сохранность данных на SSD можно не переживать с обоими экземплярами: температура у 256 ГБ модели не превышала 42 °C в пике, а у 2 ТБ – 53 °C в пике. SSD у 256 ГБ модели после заполнения сильно проседает, а вот у 2 ТБ модели SSD лучший среди всех ноутбуков, что мы тестировали.

4. Троттлинг-тесты.

На руках у меня 2 ноутбука с macOS: это Macbook Pro 13 2020 с 2 портами Thunderbolt 3 и чипом Apple M1, а также мой рабочий Xiaomi Mi Notebook Pro GTX. Как себя должен вести в троттлинг-тесте Macbook Pro 13 с 4 портами Thunderbolt 3 (2018 и 2020 годы выпуска) и чипом Intel Core i5 (8259U в случае с 2018 годом и 1038NG7 в случае с 2020 годом) я узнал, благодаря помощи Кирилла Варшамова (kavars) и Владимира Баруса. Однако, у нас же на руках Macbook Pro 13 с 2 портами Thunderbolt 3 и на новом чипе M1, так что в первую очередь логичнее его сравнивать по части охлаждения с предыдущим поколением Macbook Pro 13 с 2 портами TB3 на Core i5 8257U.



Скриншот 57. Macbook Pro 13 2019 2x TB3 (Core i5 8257U). Троттлинг-тест. Источник - notebookcheck.net.

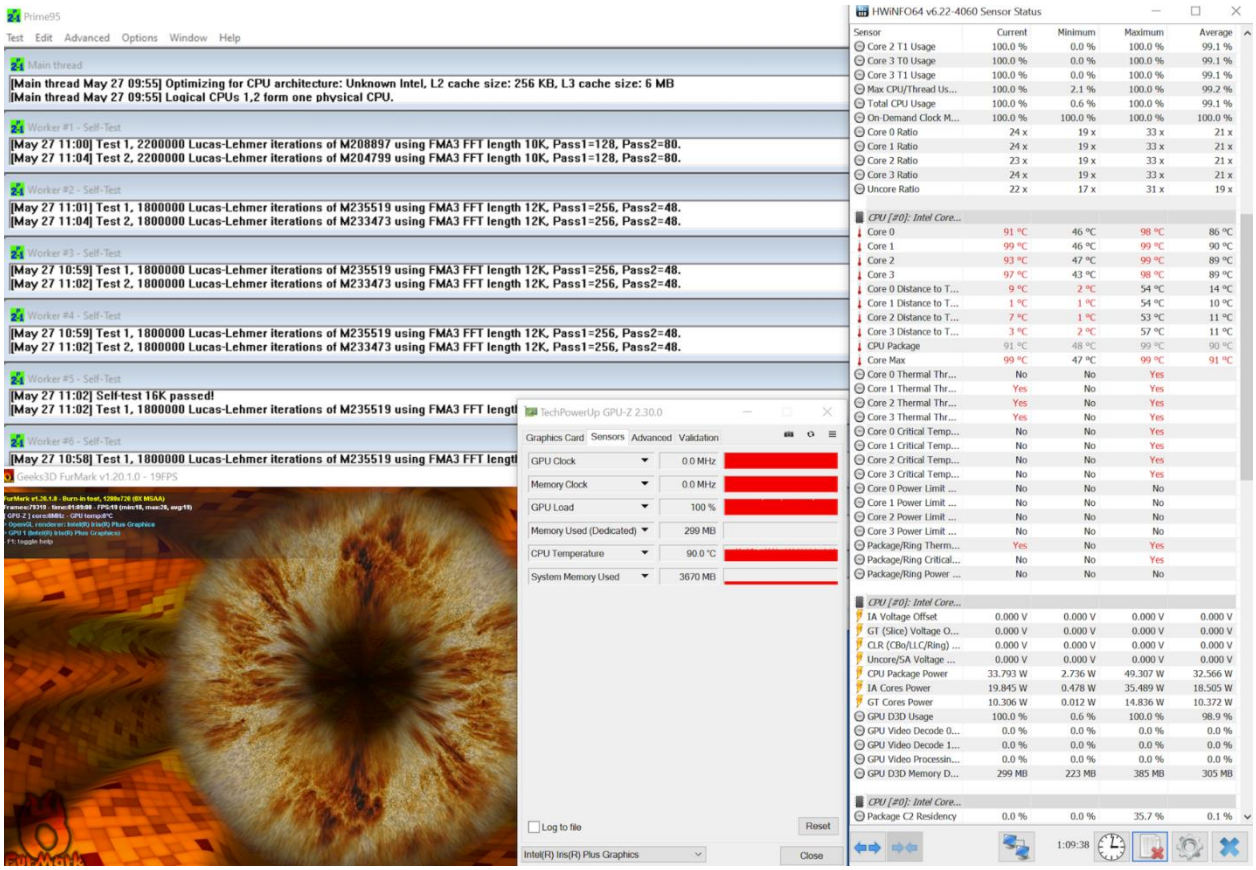
Я нашёл статью с обзором Macbook Pro 13 2019 с 2x Thunderbolt 3 (Core i5 8257U) на сайте notebookcheck.net в двух переводах (с немецкого): русский и английский (на английский автор свою статью переводил сам). В статье автор демонстрирует, что за час троттлинг-теста CPU (через Prime95) и GPU (через Furmark) его SoC Intel в среднем выделял 27 Ватт тепла (или потреблял 27 Ватт энергии), из которых 14 Ватт были отведены под

графический ускоритель. При этом максимальная температура составила 93 градуса, средняя – 89.

Вы можете возразить, что сравнивать то нужно модель на M1 с моделью на Intel с 2 портами Thunderbolt 3 2020 года, а не 2019 года, но модели 2019 и 2020 годов не отличаются друг от друга процессорами и охлаждением. Разница между ними сводится по большей части лишь к замене клавиатуры «бабочка» на старую добрую «ножницы», увеличению объёма накопителя SSD вдвое, выводу клавиши Esc с TouchBar в отдельную кнопку и т.д. В общем, сравнение с 2019 годом будет вполне релевантно.

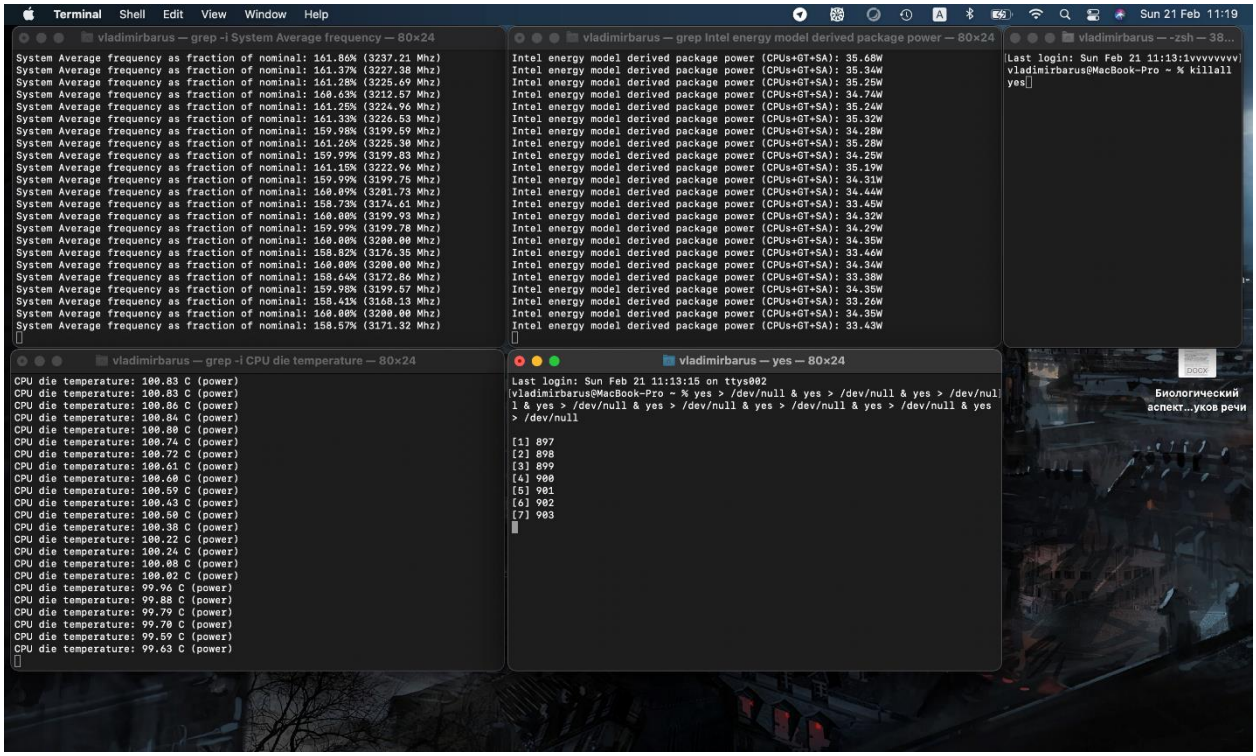
Для начала, я удивлён тем, что Apple «разгоняют» чипы Intel с TDP 15 Ватт до 27 Ватт. Да, виден максимум в 43 Ватта в статистике HWinfo, но это можно списать на короткий всплеск производительности за счёт Intel Turbo Boost PL2 (Short Power Max), и температуру в пике в 93 градуса можно также списать на Turbo Boost PL2, тогда как на постоянной основе SoC от Intel в предыдущем поколении Macbook Pro 13 2019 с 2 портами Thunderbolt 3 и чипом Core i5 работал с состоянием Turbo Boost PL 1 (Long Power Max) TDP 27 Ватт в среднем. И охлаждение, конечно, не идеально справлялось с таким тепловыделением, но, заметьте, даже в пике температура не превысила 93 градусов, в среднем составляла 89 градусов, а на момент снятия скриншота и вовсе составила 87 градусов. Да, это не хорошо, т.к. выше, чем 85 градусов, значит более быстрая деградация процессора под нагрузкой со временем имеет место из-за перегрева (см. видео «Поговорим про деградацию и БУ товары, которые выдают за новые» на канале у Pro Hi-Tech с 7:53).

Сперва удостоверимся в том, что показаниям автора можно верить. Поскольку предыдущего Macbook Pro с 2 портами Thunderbolt 3 у меня на руках нет, мы будем сопоставлять данные с Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (Core i5 1038NG7), которые нам предоставил Владимир Барус, с данными из его статьи с обзором: немецкий (оригинал), английский (авторский перевод), русский (перевод от Антона Авдюшкина).



Скриншот 58. Macbook Pro 13 2020 4x TB3 (Core i5 1038NG7). Троттлинг-тест. Источник - notebookcheck.net.

Судя по показаниям автора, за час тестирования у него чип Intel в Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (i5 1038NG7) нагрелся в пике до 99 градусов при пиковом тепловыделении 49 Ватт. В среднем температура у него составляла 91 градус, а тепловыделение – 32.5 Ватт.



Скриншот 59. Macbook Pro 13 2020 4x TB3 (Core i5 1038NG7). Троттлинг-тест. Данные Владимира Баруса.

Владимир Барус предоставил данные только за 5 минут стресс-теста. Длительный стресс-тест на его рабочем ноутбуке он проводить не стал, т.к. ему с этим ноутбуком ещё работать, и он бы не хотел, чтобы из-за перегрева какие-либо компоненты внутри ноутбука износились или и вовсе вышли из строя. Тем не менее, этих 5 минут хватило, чтобы SoC от Intel при работе в режиме Turbo Boost перешёл из состояния PL2 в состояние PL1 и тепловыделение ноутбука снизилось до 33.5 Ватт, что примерно соответствует тому, что мы видели в данных Andreas Osthoff в его статье на notebookcheck, исходя из чего я делаю вывод, что его данным доверять можно.

Стресс-тест мы проводили по довольно простой методике. Запускам Terminal и вводим в него X раз через пробел следующую команду:

```
yes > /dev/null &
```

Каждая такая команда даёт нагрузку в 100% на 1 поток процессора. Соответственно, чтобы нагрузить 8 потоков процессора, нам нужно дать 8 таких параллельных команд:

```
yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &
```

Чтобы прервать стресс-тест нужно ввести одну команду:

```
killall yes
```

У Владимира на скриншоте эта команда прописана в верхнем правом окне Terminal, но не введена, т.е. Enter не нажат, и команда не отправлена. Можете аналогичным образом держать у себя наготове команду для приостановки всех процессов yes и просто нажать Enter в нужный момент, чтобы прервать стресс-тест.

Для мониторинга частоты процессора на Mac с Intel мы задействовали следующую команду:

```
sudo powermetrics --samplers cpu_power 2>&1 | grep -i "System Average frequency"
```

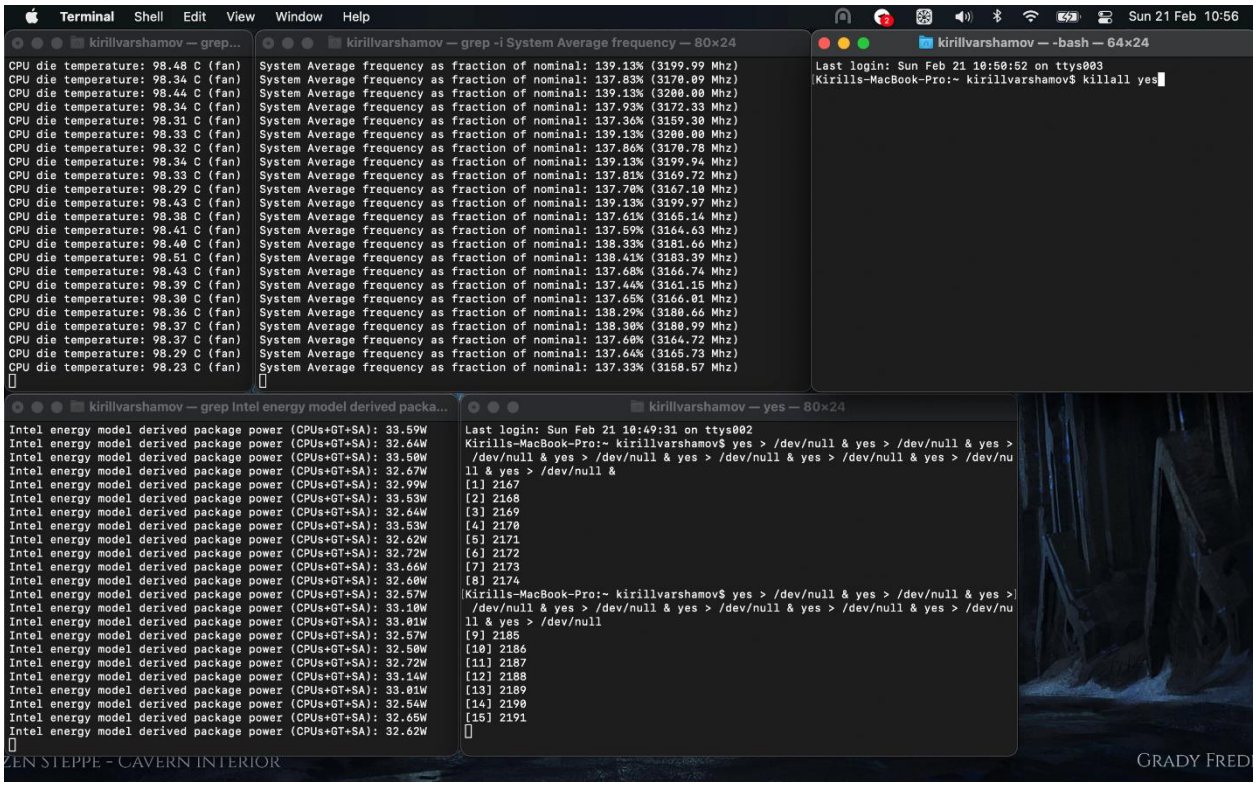
Для мониторинга энергопотребления/тепловыделения SoC от Intel мы задействовали следующую команду:

```
sudo powermetrics --samplers cpu_power 2>&1 | grep -i "package power"
```

И ниже следует команда для мониторинга температуры на Mac с Intel:

```
sudo powermetrics --samplers smc 2>&1 | grep -i "CPU die temperature"
```

Почему я говорю, что это затрагивает только Mac с Intel? Да потому, что команды для Mac с M1 будут другие. Ну и, в добавок, температуру через Terminal на компьютерах Mac с чипом M1 мы отслеживать не смогли, так что для этих целей в случае с Mac на M1 мы использовали TG Pro и iStat Menus.



Скриншот 60. Macbook Pro 13 2018 4x TB3 (Core i5 8259U). Тротлинг-тест. Данные Кирилла Варшамова (kavars).

Я также попросил Кирилла провести этот же тест, хотя бы в течение 5 минут, на его Macbook Pro 13 2018 с 4 портами Thunderbolt 3 (Core i5 8259U) и у него картина примерно та же: частота около 3.2 ГГц, температура 98 градусов, а разницу в 2 градуса можно списать как-раз на разницу в тепловыделении в 1 Ватт (32.5 Ватта вместо 33.5).

А выводы по охлаждению у меня печальные для владельцев Macbook Pro с 4 портами Thunderbolt 3 (будь то модель 2018 года или 2020 года):

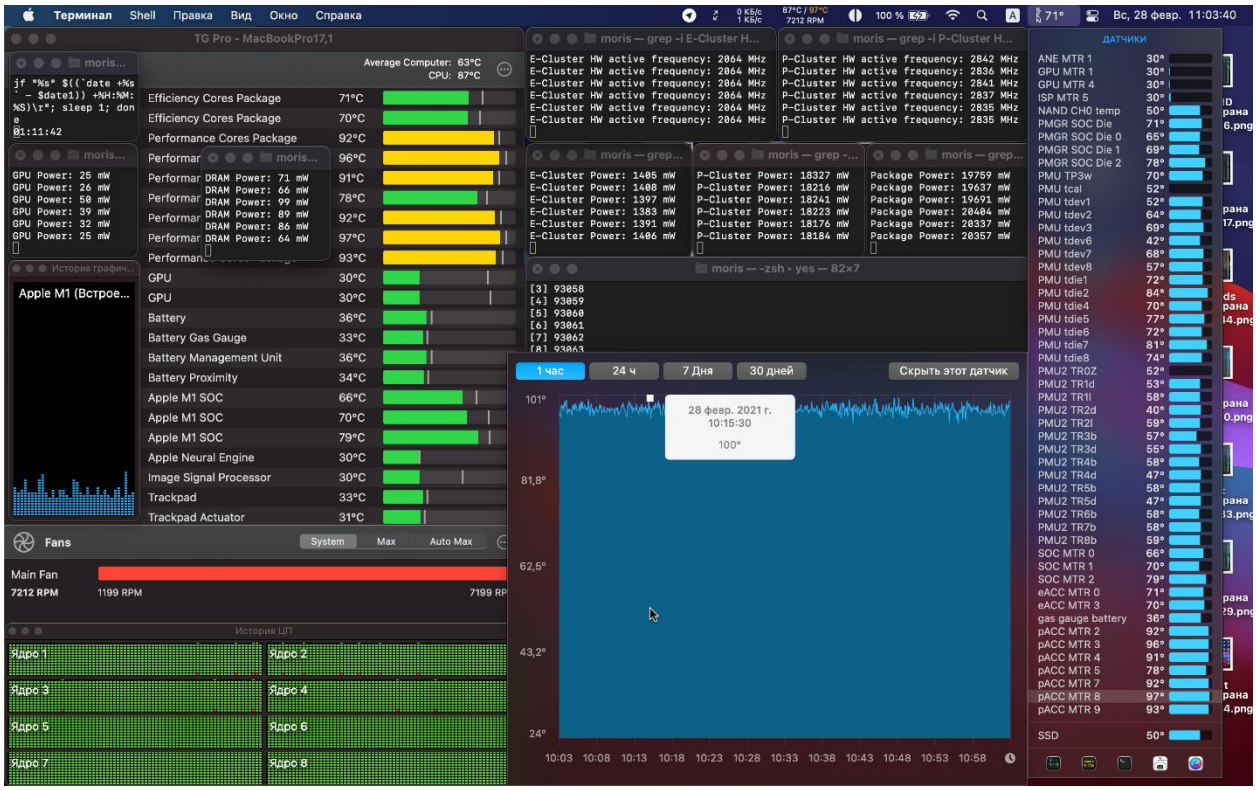
Apple просто подняли в них TDP с 27 Ватт до 32.5 Ватт, а охлаждение то там не сильно эффективнее, чем в модели с 2 портами Thunderbolt 3, из-за чего модели с 4 портами Thunderbolt 3 греются намного сильнее. Да и переход с 14 нм на 10 нм в Macbook Pro 13 2020 не особо то и ощутим: температуры те же, тепловыделение то же, частота примерно та же. Только архитектурные изменения могут как-либо повлиять на разницу в производительности между этими двумя макбуками.

Вы можете возразить, что нельзя сравнивать данные в стресс-тестах, где задействуются только процессорные ядра и кэш с данными, где задействуется ещё и графический ускоритель. Тут я отвечу: у чипов Intel нет отдельного ограничения по TDP для процессорной части, графической части, нейронным блокам и т.д. У них TDP единый для всего SoC. Вы можете прочитать наш обзор на Asus Zenbook Flip S, у которого имеется лишь интегрированная графика, и там Вы увидите, что при задействовании что только процессора, что процессора вместе с интегрированной графикой, общее тепловыделение всего SoC остаётся одним и тем же, т.к. лимит по TDP общий для всех компонентов SoC от Intel. Так что, если мы задаёмся вопросом, с каким предельным тепловыделением работают SoC от Intel, одной только команды yes для стресс-теста более чем достаточно. Вот если мы хотим ещё и выявить, насколько снижается частота процессора, когда SoC необходимо ещё выделить энергию помимо него и на графический ускоритель, тут да, нужно ещё грузить графику.

Итак, мы узнали, с каким тепловыделением и с какой температурой функционировал SoC в предыдущем Macbook Pro 13 с 2 портами Thunderbolt 3 (Core i5 8257U), мы удостоверились, что данным автора Andreas Osthoff с сайта notebookcheck более-менее верить можно, сравнив данные с другой его статьи с обзором Macbook Pro 13 с 4 портами Thunderbolt 3 (1038NG7), с тем, что имеется на руках у Владимира Баруса. Мы также выяснили, что разницы между 2020 и 2018 годами по частью охлаждения нет практически никакой.

Осталось выяснить, изменилось ли что-либо в случае с Macbook Pro с чипом Apple M1. Стало ли лучше, хуже, или осталось всё по-прежнему?

И спойлер: чёрт тебя дери, Apple, стало хуже. Apple умудрились спроектировать великолепный, с инженерный точки зрения, чип на ARM, но обо**ались с проектированием охлаждения под этот самый чип в ноутбуке.



Скриншот 61. Macbook Pro 13 2020 (Apple M1). Стресс-тест командой yes в 8 потоков. Спустя час.

Спустя час стресс-теста я снял скриншот с ноутбука, и Вы видите здесь, что при тепловыделении в 20 Ватт (не 27 Ватт и не 33 Ватта) ноутбук в течение стресс-теста нагревался до 100 градусов при том, что его кулеры крутились на максимум (это видно в окне TG Pro слева). Какого чёрта, Apple? В прошлом поколении Ваш же ноутбук с 2 портами Thunderbolt 3, хоть и худо-бедно, всё же справлялся с охлаждением процессора при его тепловыделении в 27 Ватт, а в Macbook с чипом M1 ноутбук категорически не справляется с охлаждением чипа с тепловыделением 20 Ватт. Как так-то?

Выше Вы видите несколько открытых окон в Terminal. Ниже я распишу, в чём роль каждого окна, какие команды там выполняются.

1. Верхний левый угол. Секундомер. Да, обычный секундомер, чтобы я не отсчитывал время сам, а точно знал, сколько времени прошло со старта стресс-теста. Запускается довольно простым набором команд

```
date1=`date +%s`; while true; do days=$(( $(($(date +%s) - date1)) / 86400 ));
echo -ne "  $days day(s) $(date -ujf "%s" $((`date +%s` - $date1)) +%H:%M:%S)\r";
sleep 1; done
```

2. Верхний левый угол, ниже секундомера. Энергопотребление / тепловыделение графического ускорителя. Проверяем следующей командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "GPU Power"
```

3. Верхний левый угол, правее мониторинга GPU. Энергопотребление / тепловыделение объединённой памяти. Проверяем следующей командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "DRAM Power"
```

4. Верх, центр. Тактовая частота кластера из 4 энергоэффективных ядер процессора. Проверяем следующей командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "E-Cluster HW active frequency"
```

5. Верх, чуть правее центра. Тактовая частота кластера из 4 высокопроизводительных ядер процессора. Проверяем следующей командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "P-Cluster HW active frequency"
```

6. Верх, центр, чуть ниже мониторинга частота. Энергопотребление / тепловыделение кластера энергоэффективных ядер. Проверяем командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "E-Cluster Power"
```

7. Верх, центр, чуть правее тепловыделения энергоэффективных ядер. Проверяем тепловыделение высокопроизводительных ядер процессора:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "P-Cluster Power"
```

8. Верх, центр, правее тепловыделения производительных ядер. Проверяем энергопотребление / тепловыделение всего SoC этой командой:

```
sudo powermetrics –samplers cpu_power 2>&1 | grep -i "Package Power"
```

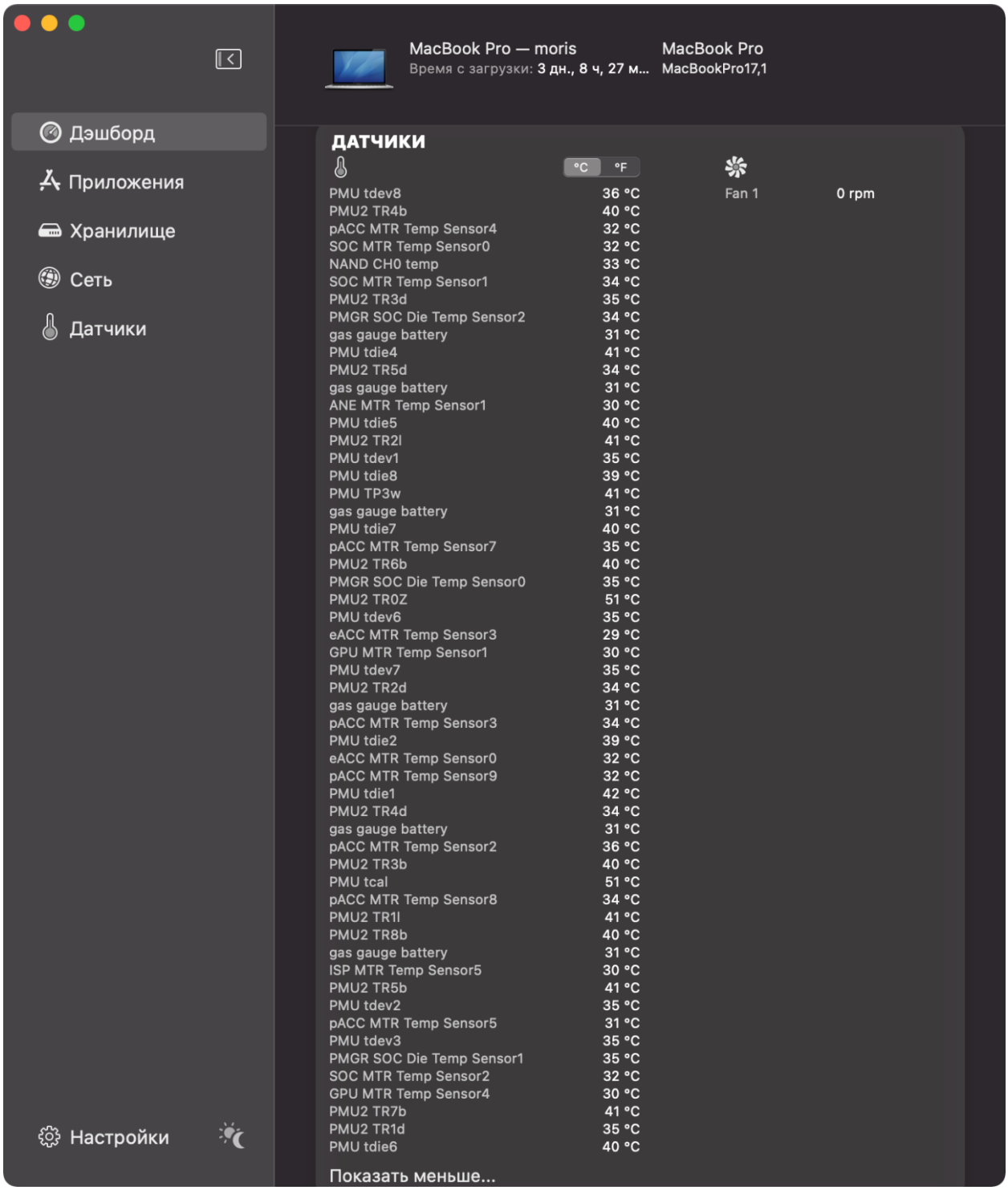
9. Центр. Просто команда yes для стресс-теста CPU в 8 потоков:

```
yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &  yes > /dev/null &
```

Слева по центру и в левом нижнем углу Вы также можете видеть историю графического ускорителя и историю ЦП, которые Вы можете отобразить, нажав на «Окно» в Меню Баре приложения «Мониторинг Системы». Слева, помимо этого, Вы видите мониторинг сенсоров в TG Pro.

Также справа Вы видите мониторинг сенсоров с историей температуры одного из высокопроизводительных ядер процессора в iStat Menus.

Данные с сенсоров в Terminal обновляются раз в 5 секунд, данные с сенсоров в TG Pro и iStat Menus – раз в 4 секунды.

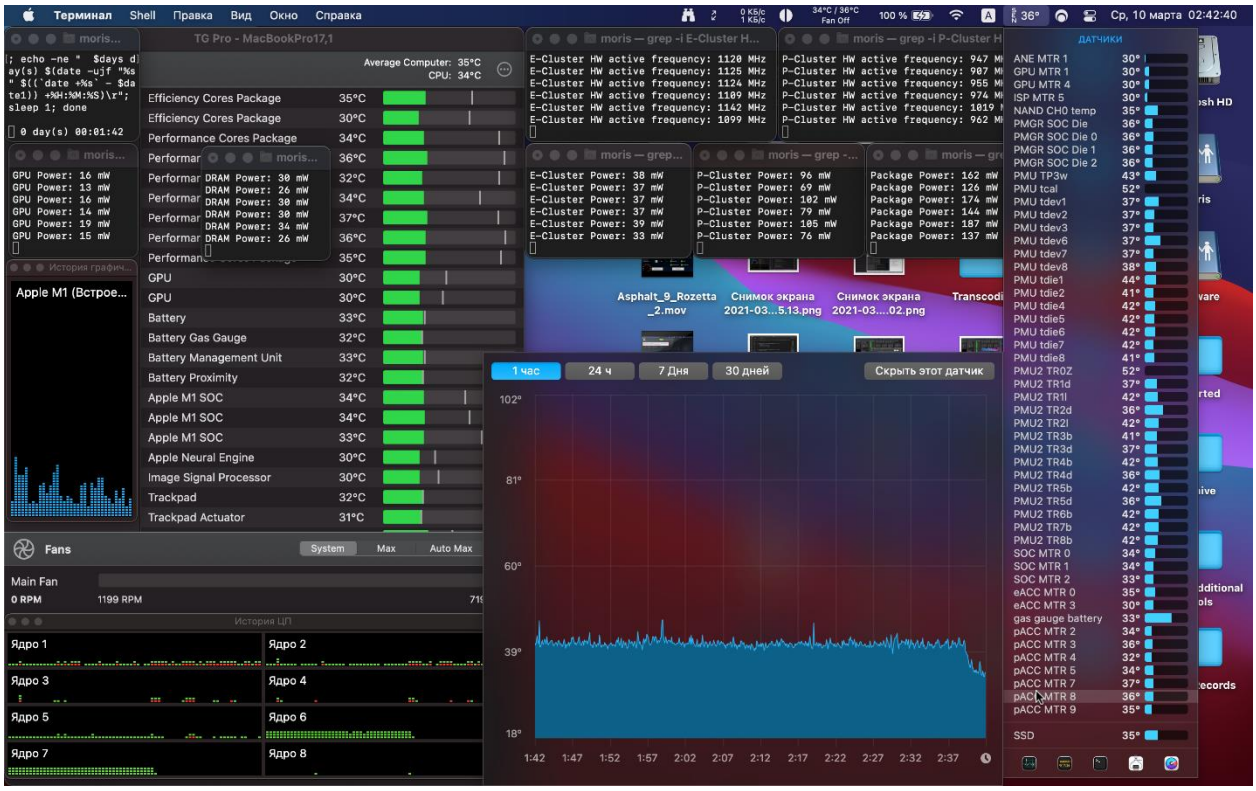


Скриншот 62. Macbook Pro 13 2020 (Apple M1). Информация о температуре с сенсоров из iStatistica.

Некоторые могут задаться вопросом, отчего же я не воспользовался iStatistica на Mac вместо Tunabelly TG Pro и iStat Menus. Да я бы, может, и воспользовался, если бы в iStatistica сенсоры не были расставлены вразброс,

т.е. сложно понять, какие сенсоры к какой группе относятся (например, какие к процессору, какие к SSD, и т.д.). В добавок тут нет никаких графиков, и в текущей версии (5.0.1) отсутствует также и возможность экспортировать данные за время их замера через iStatistica, так что и после нам в Excel графики не построить. Не пойдёт.

Т.к. я не нашёл полноценной замены HWinfo для macOS, которая позволила бы вывести график на основе наибольшей температуры с сенсоров, зафиксированной в течение каждой секунды их опроса, и отобразить статистику за всё время тестирования, я и использую то, что есть. А из того, что есть, связку из Tunabelly TG Pro и iStat Menus, а также мониторинга частоты и тепловыделения в Terminal, я и счёл наиболее подходящим решением.



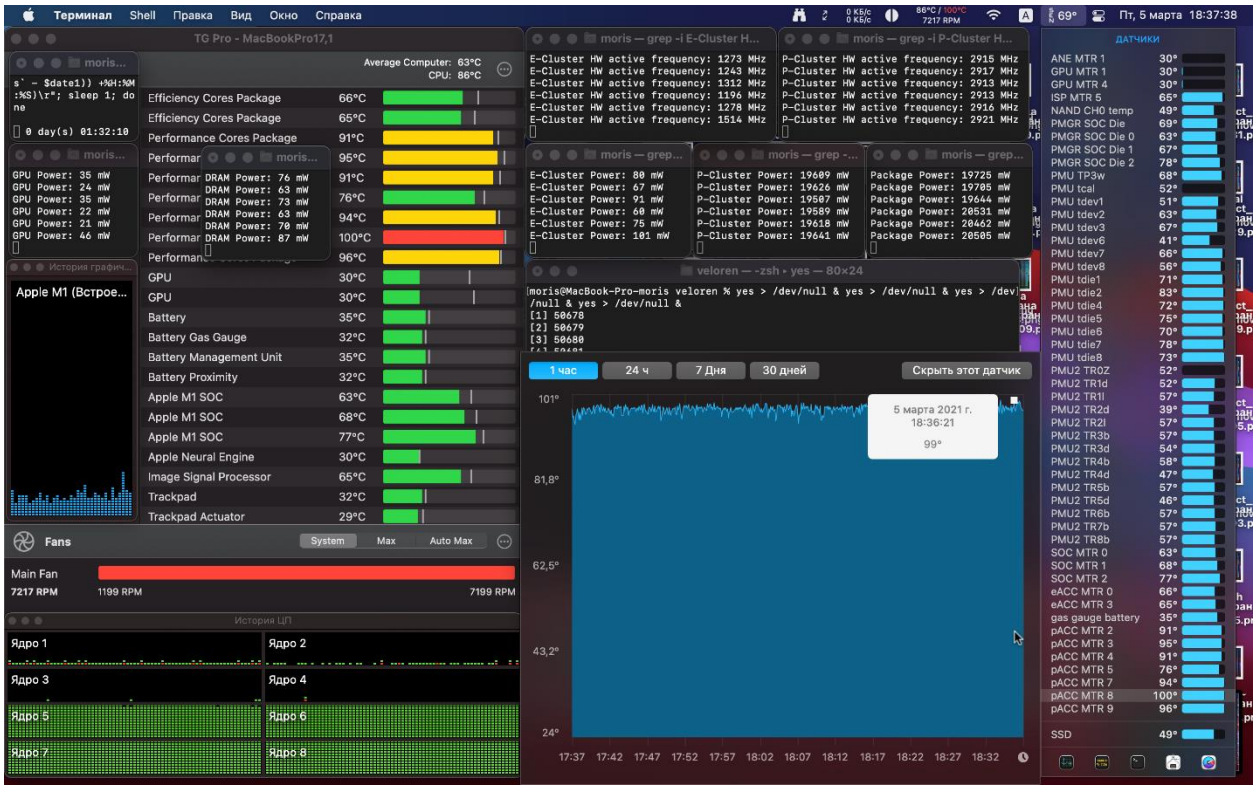
Скриншот 63. Macbook Pro 13 2020 (Apple M1). Мониторинг энергопотребления в простое.

Если у Вас есть сомнения по поводу того, не влияет ли мой мониторинг существенно результаты, можете обратить внимание на скриншот выше. До снятия скриншота я закрыл открытое ранее окно Telegram с циклически воспроизводящимся видео в одном чате, отсюда и резкое падение температуры с 42 градусов до 35 в среднем. Энергопотребление SoC M1 в

простое с учётом моего мониторинга составляет всего... менее чем 200 мВт... Вы где-нибудь видели ноутбук, который в простое потребляет столь мало? Оставим это до главы про автономность. Влияние мониторинга на результаты стресс-теста чрезвычайно мало, так что его мы не учитываем.

Итак, возвращаясь к теме: был Macbook Pro 13 2019 (он же 2020 с заменой «бабочки» на «ножницы») с 2 портами Thunderbolt 3 и SoC Intel Core i5 8257U, который худо-бедно справлялся с охлаждением чипа, с тепловыделением 27 Ват под макс. нагрузкой. И есть новый Macbook Pro 13 2020 с SoC Apple M1, который уже не справляется и с охлаждением 20 Ватт. У Macbook Pro с чипом Apple M1 циклически скачет тепловыделение SoC при стресс-тесте в диапазоне 19.5-20.5 Ватт, но в среднем мы имеем 20 Ватт.

Apple, как это объяснить? Т.е. если новый Macbook стал сильно эффективнее предыдущих, можно совсем наплевать на его охлаждение?

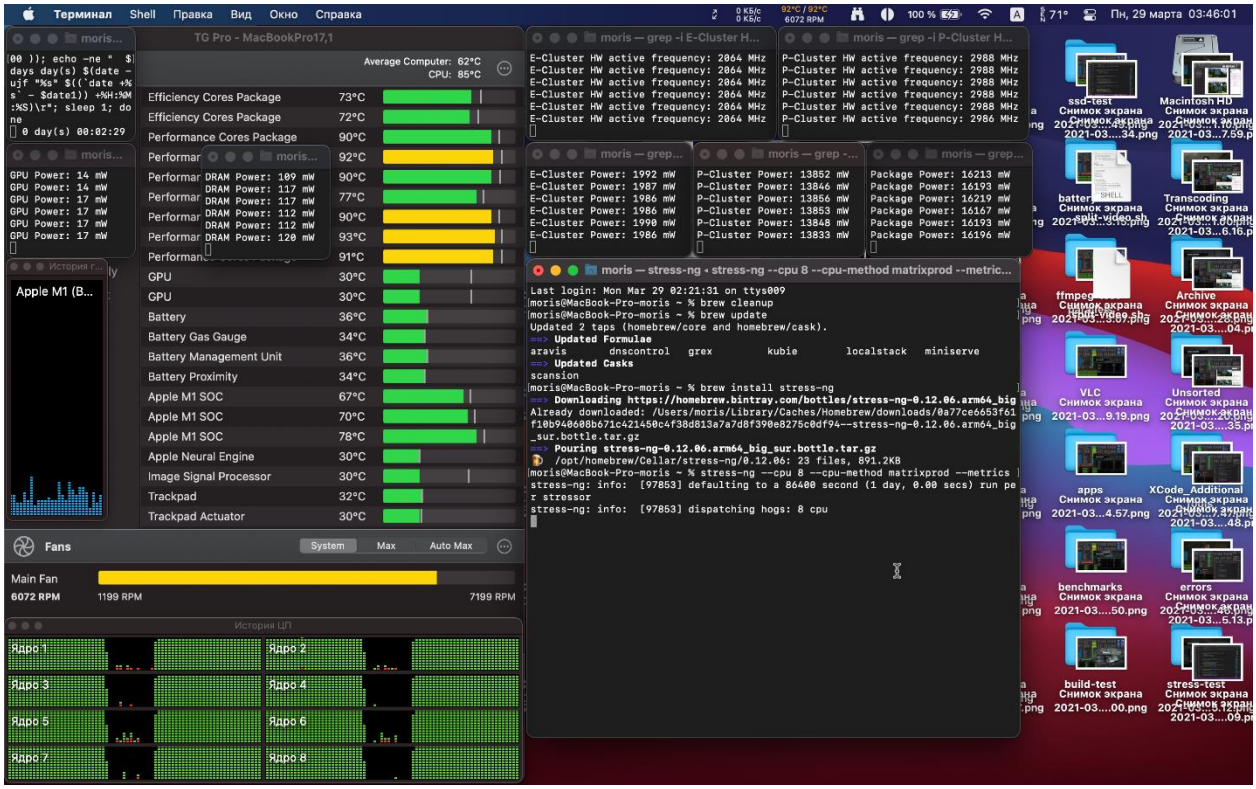


Скриншот 64. Macbook Pro 13 2020 (Apple M1). Стресс-тест командой yes в 4 потока. Спустя 1.5 часа.

Если запустить 4 потока команды yes, то будут загружены только 4 высокопроизводительных ядра, а не все 8, включая энергоэффективные. И в таком случае ничего особо не изменится. Мы видим, что энергопотребление / тепловыделение процессора также скачет в диапазоне 19.5-20.5 Ватт и в

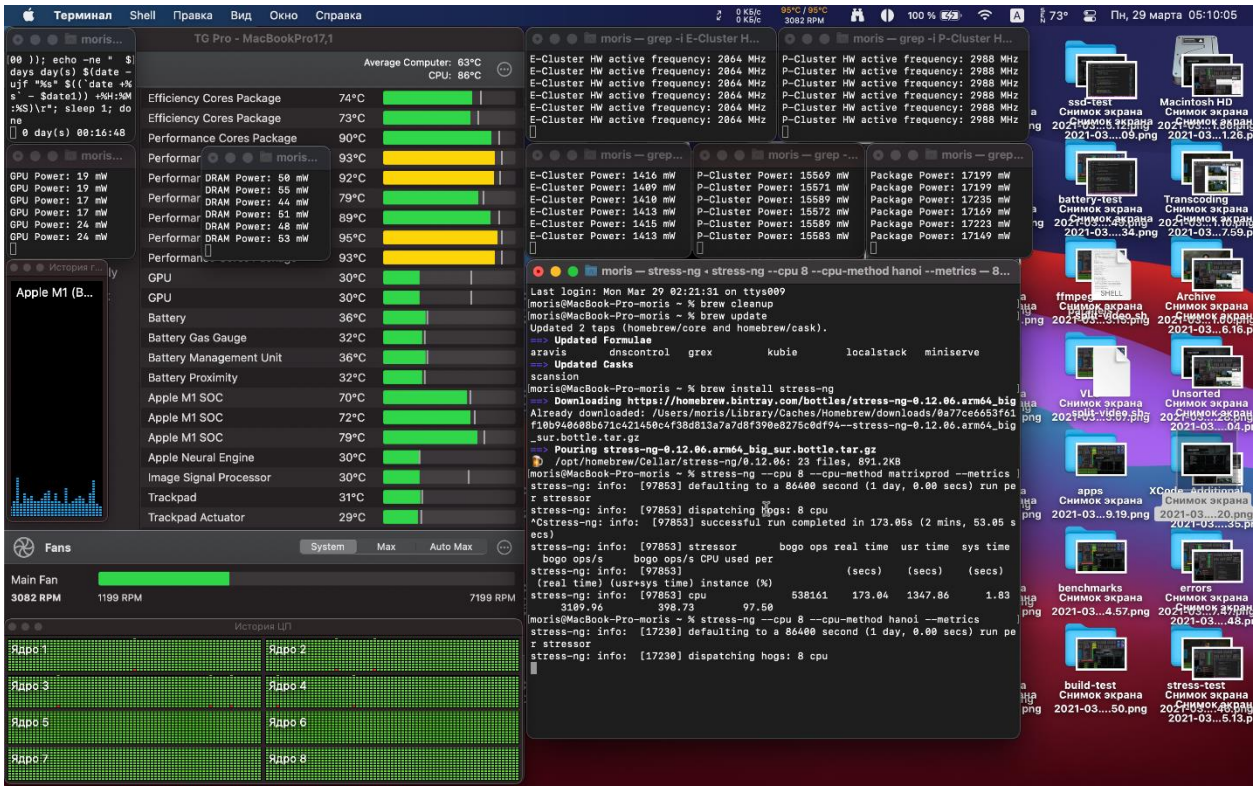
среднем составляет около 20 Ватт. Единственное, что изменилось, так это то, что за счёт сокращения расхода энергии на энергоэффективные ядра на 1.5 Ватта выросла тактовая частота высокопроизводительных ядер на... целых 80 МГц. Короче говоря, разницы практически нет.

Занятно, конечно, что энергоэффективные ядра потребляют столь мало, всего 1.4 Ватта в стресс-тесте. Это очень круто и позже в главе с анализом автономности ноутбука мы вернёмся к этому вопросу, а пока продолжим стресс-тест. Если Вы читали мой обзор на Asus ROG Strix G15, у Вас может возникнуть вполне логичный вопрос: а почему же я не воспользовался утилитой stress-ng для стресс-теста? Она же запросто ставится через brew.



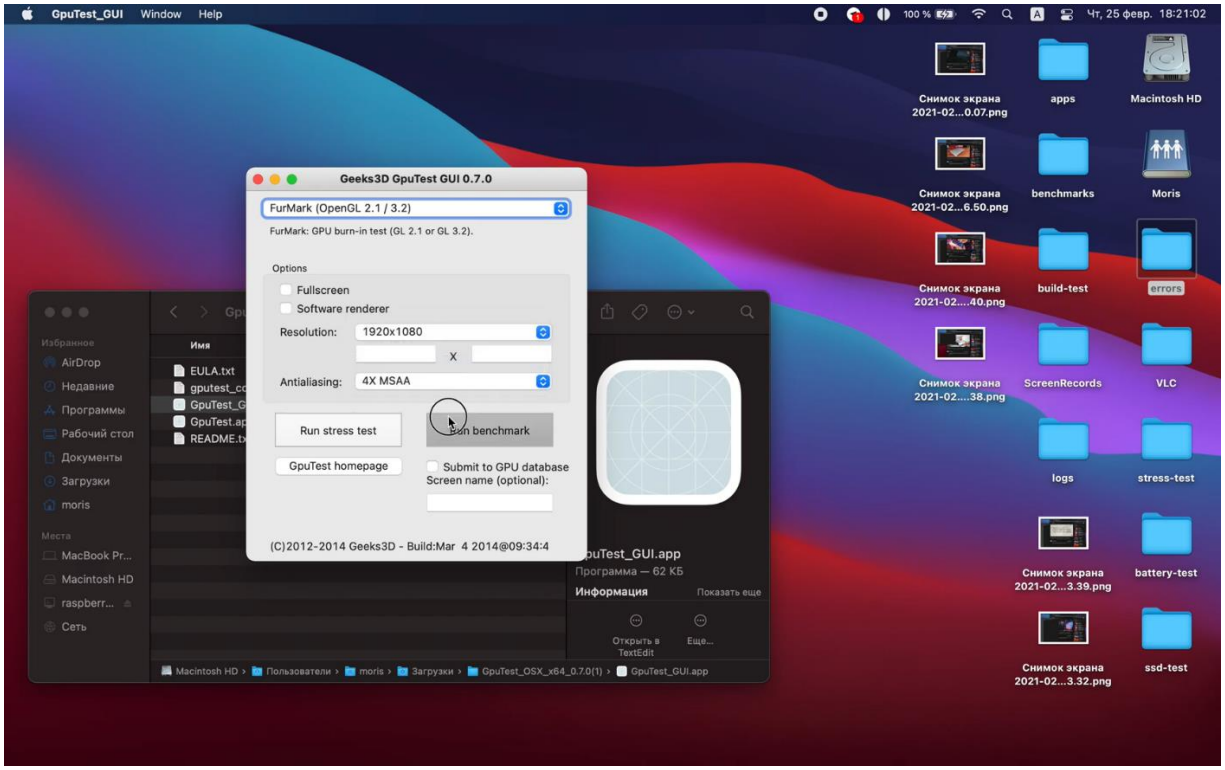
Скриншот 65. Стресс-тест CPU утилитой stress-ng. Метод matrixprod.

В stress-ng метод matrixprod, который я использовал для стресс-теста Asus, и с помощью которого я проводил тестирование своей «малинки» (одноплатного микрокомпьютера Raspberry Pi 4), попросту не смог полноценно нагрузить Macbook Pro с SoC Apple M1. Энергопотребление и тепловыделение Apple M1 при запущенном stress-ng составляли всего 16 Ватт. Намного меньше, чем те 20 Ватт, которые мы видели ранее.



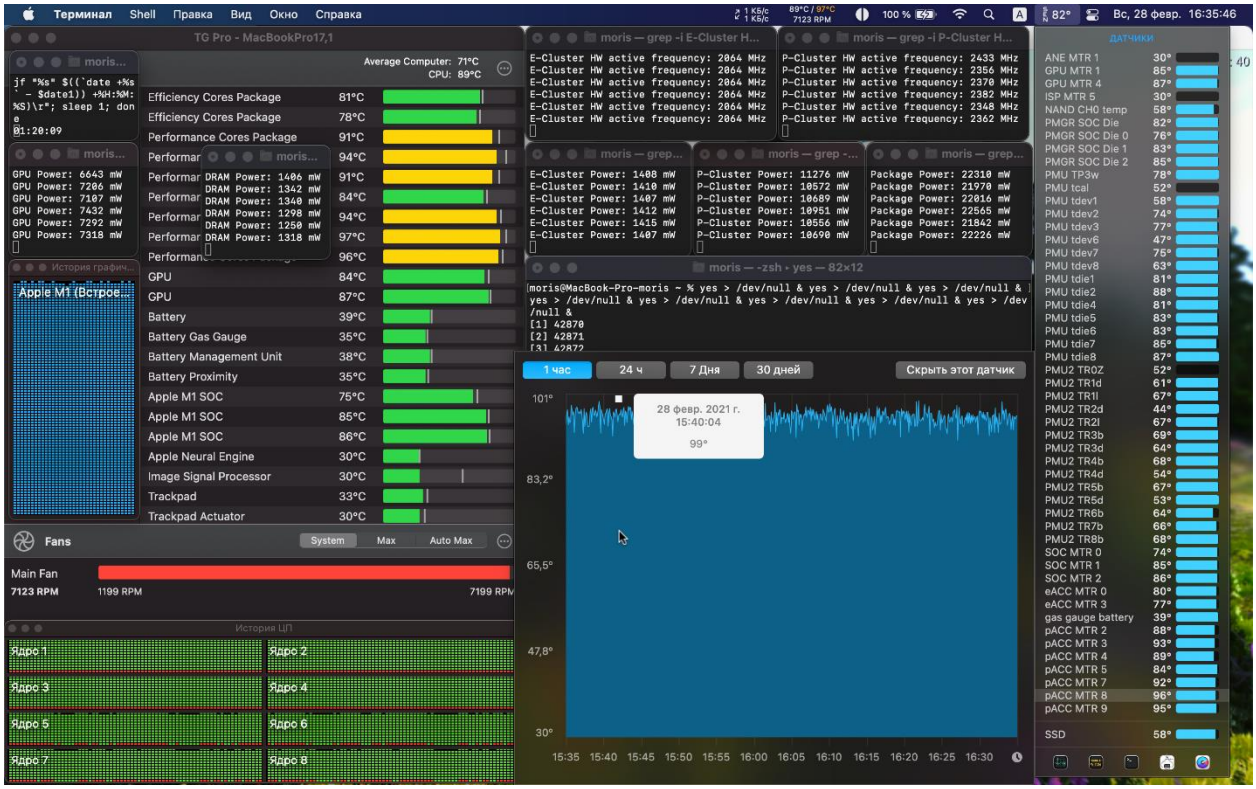
Скриншот 66. Стресс-тест CPU утилитой stress-ng. Метод hanoi.

Я затем начал перебирать все остальные методы, но больше всего энергии в stress-ng по итогу потреблял только hanoi, 17 Ватт, а у gamma, idxt, ipv4checksum, как и у matrixprod заставили M1 потреблять лишь 16 Ватт. Других толковых решений под macOS, кроме yes, не нашёл. Ладно, нагрузим теперь графический ускоритель и посмотрим, что у него там с троттлингом.



Видео 3. Невозможность запуска стресс-тестов в GpuTest на Macbook Pro 13 2020 (Apple M1).

Я хотел при помощи утилиты GpuTest запустить тот же стресс-тест Furmark, который мы используем для стресс-теста графического ускорителя на ноутбуках с ОС Windows, но он попросту отказался работать на macOS 11 Big Sur независимо от того, какие настройки графики я задавал при запуске. Скорее всего, дело тут в том, что Apple отказались от поддержки OpenGL ещё 2 года назад, так что странно удивляться тому, что утилиты, всё ещё задействующие API OpenGL теперь местами (или полностью) сбоят.



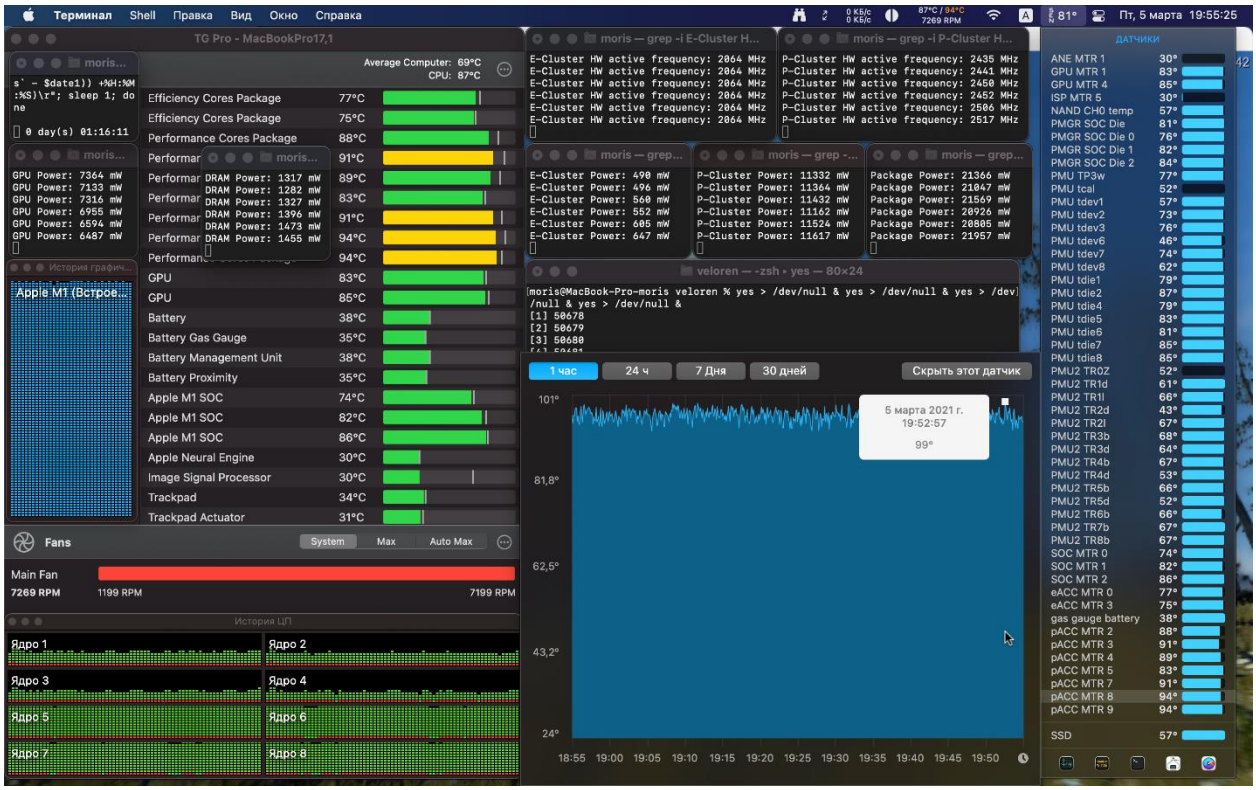
Скриншот 67. Macbook Pro 13 2020 (Apple M1). Стресс-тест CPU (yes x8) и GPU (Unigine Valley). Спустя 1.33 часа.

К сожалению, нормальных стресс-тестов GPU, не ограниченных во времени (нам же нужно циклически грузить GPU) и использующих API Metal, в природе не существует под macOS, поэтому вариантов для стресс-теста 2: воспользоваться Unigine Valley со сглаживанием x4 (с x8 имеем чёрный экран) или найти бенчмарк в какой-нибудь игре, который циклически воспроизводится и не прерывается после получения результата (на 2-ой вариант я наткнулся случайно, о нём позже). Опробуем Unigine Valley.

Как видите, энергопотребление и тепловыделение немного выросло за счёт того, что теперь была загружена ещё и объединённая память (DRAM), и её потребление составило 1.3-1.4 Ватта. На примерно это же значение выросло энергопотребление всего SoC (вкл. DRAM). В целом картина

осталась всё та же. Ноутбук как не справлялся с охлаждением, так и не справляется. Что занимательно: GPU у нас потребляет в пике до 7.5 Ватт при том, что демонстрирует весьма внушительную производительность.

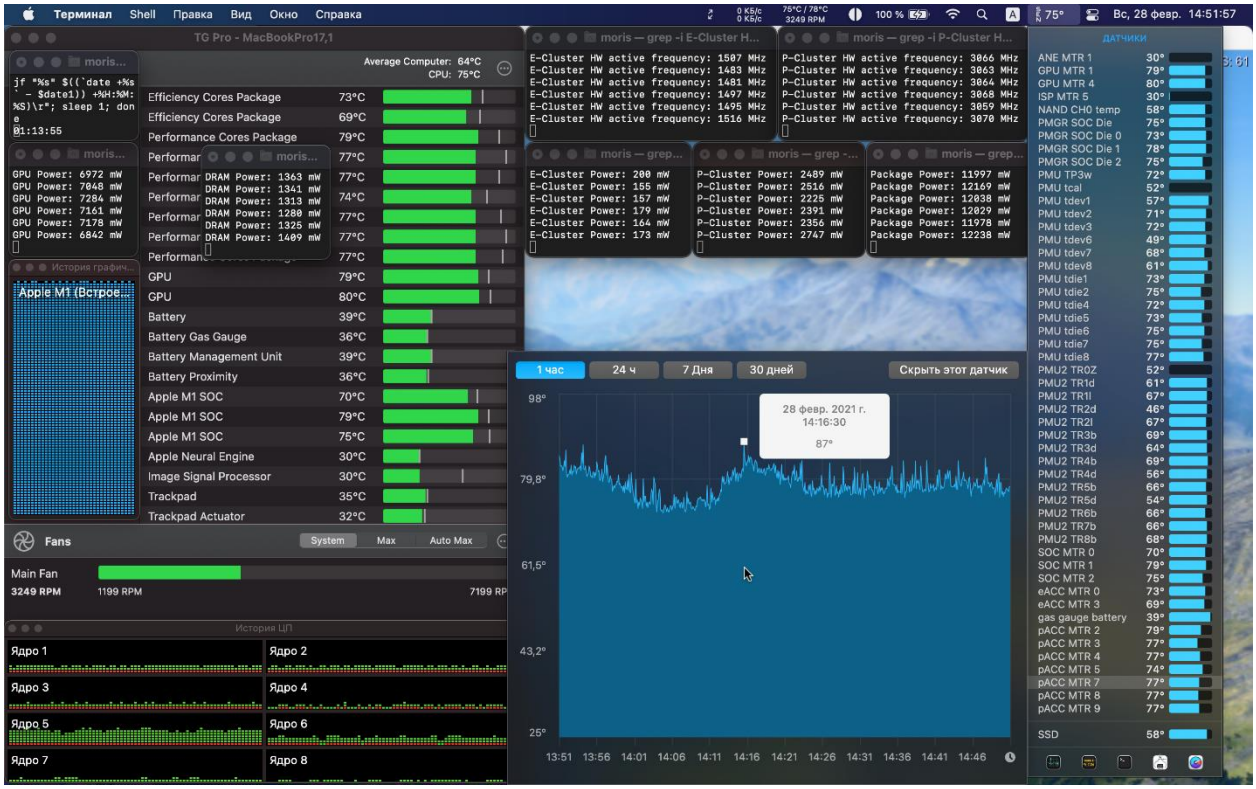
То, что энергопотребление и тепловыделение DRAM существенно не влияют на температуру всего чипа вполне закономерно, т.к. объединённая память, хоть и припаяна вплотную к самому SoC, но, всё же, является отдельным компонентом, тепловыделение в 1.5 Ватта которого вполне может быть рассеяно пассивно прямо под крышкой самой памяти, даже никак не влияя на нагрев остальных компонентов в SoC, потому и учитывать её при ограничении TDP SoC не имеет смысла. Нет смысла ограничивать TDP из-за того, что на нагрев никак не повлияет, разве что минимизации энергопотребления ради. Тут Apple поступили правильно, но я всё ещё не понимаю, из-за чего же они не установили в этот ноутбук нормальное охлаждение, которое легко бы справилось с тепловыделением в 20 Ватт. Что пошло не так, Apple?



Скриншот 68. Macbook Pro 13 2020 (Apple M1). Стресс-тест CPU (yes x4) и GPU (Unigine Valley). Спустя 1.25 часа.

Если не грузить энергоэффективные ядра процессора, а грузить только высокопроизводительные ядра процессора и интегрированный графический ускоритель, дав команду yes всего на 4 потока, а не на 8, никакой

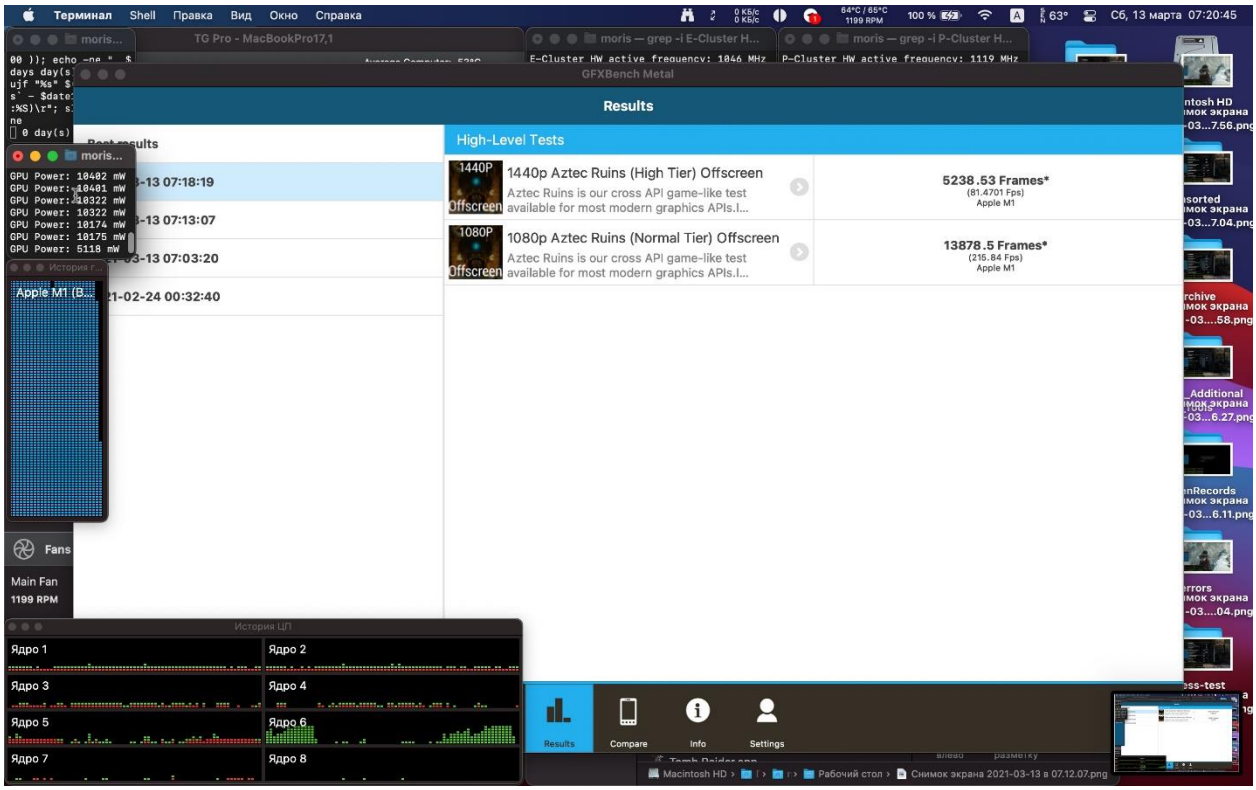
существенной разницы с тем, что Вы видели ранее, не будет. Я всё ещё не понимаю, что мешало Apple просто поставить сюда то же охлаждение, которое было установлено в предыдущем поколении Macbook Pro 13 с 2 портами Thunderbolt 3. Не было бы никаких вопросов с температурой.



Скриншот 69. Macbook Pro 13 2020 (Apple M1). Стресс-тест GPU (Unigine Valley). Спустя 1.2 часа.

В Unigine Valley при нагрузке только на видеокарту, её энергопотребление / тепловыделение составили всё те же 7.5 Ватт в пике. Температура SoC при этом держалась во время теста в пределах 75-80 градусов большую часть времени, за исключением редких всплесков до 83-85 градусов и разового скачка температура до 87 градусов. Кулеры при этом крутились чуть менее чем в пол оборота.

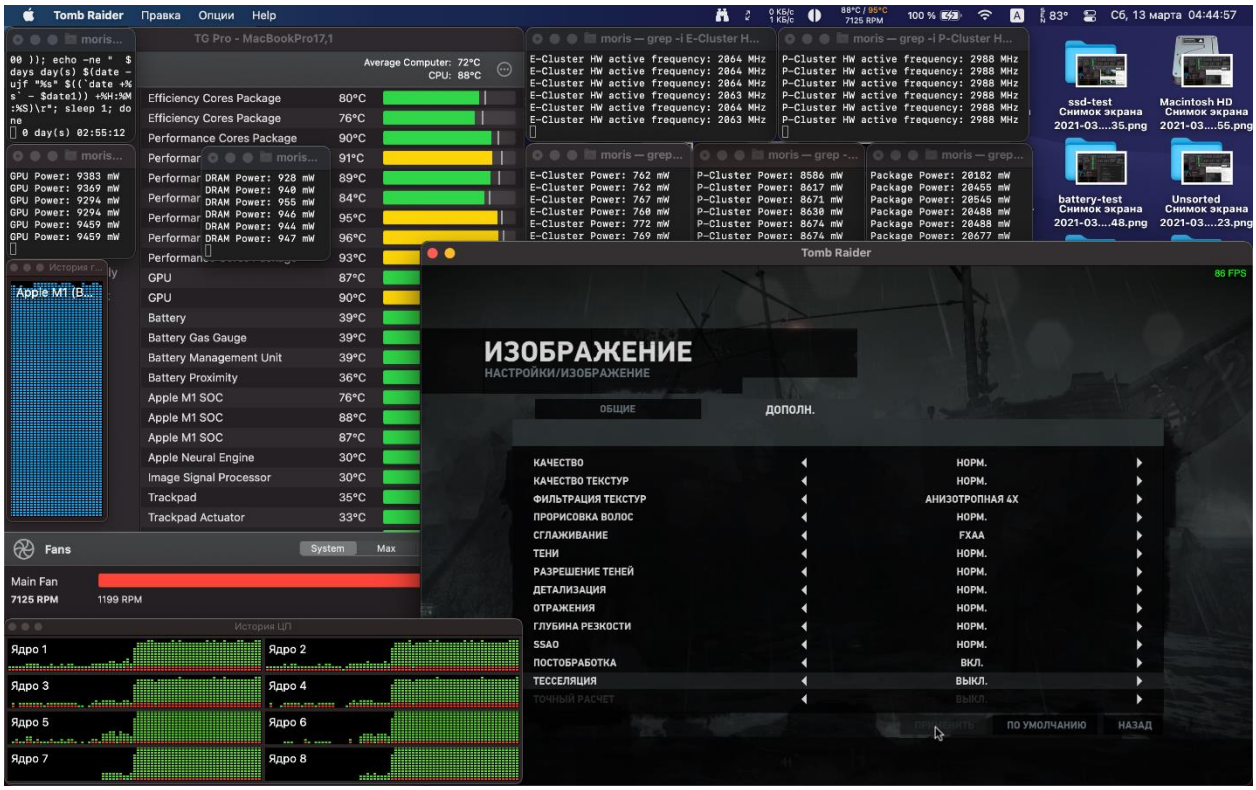
Но вот нюанс. Я сейчас Вам заспойлерю один графический тест, который далее будет в 6-ой главе.



Скриншот 70. Macbook Pro 13 2020 (Apple M1). Тест сцены Aztec Ruins в GFXBench. API Metal.

Привожу Вам один из тестов в GFXBench, который я проводил в режиме Offscreen (в таком случае нет ограничения FPS до 60, т.е. до частоты обновления картинки на экране) из-за вертикальной синхронизации. Как Вы видите, в сцене Aztec Ruins, отрисованной при помощи API Metal, энергопотребление / тепловыделение интегрированного графического ускорителя в Apple M1 достигло 10.4 Ватт в пике. Т.е. те 7.5 Ватт, что мы видели ранее в стресс-тесте Unigine Valley это на деле не полная загрузка видеоадаптера и его можно загрузить сильнее. Но где же найти стресс-тест, который будет циклически рендерить сцены, которые будут на все 100% грузить видеоускоритель? В общем-то нигде, таких нет для macOS.

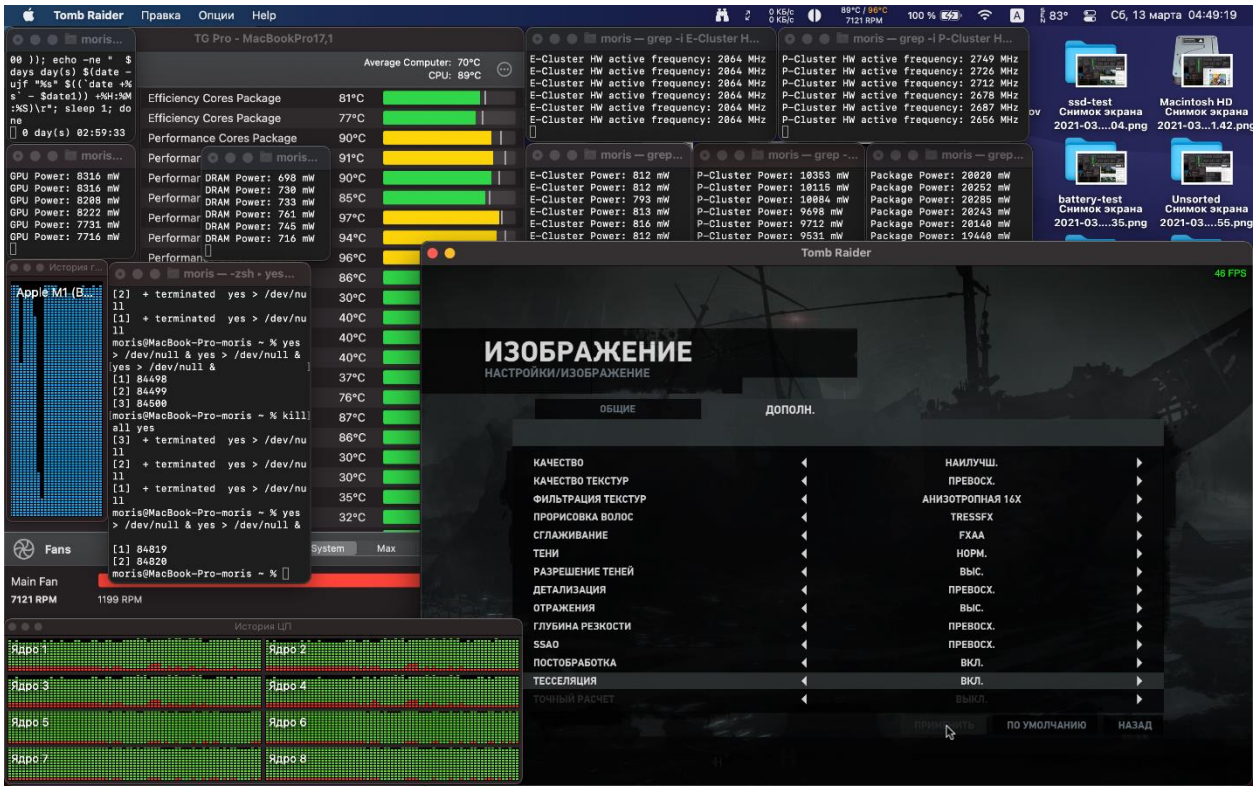
Я загрузил себе консольный phoronix test suite, пробовал ставить тесты оттуда и запускать их, но постоянно сталкивался с проблемами их совместимости с macOS. И что тогда делать? Решение нашлось, конечно, но оно далеко не идеальное.



Скриншот 71. Macbook Pro 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре.

Решением стала панель настроек графики в игре Tomb Raider (2013), обновлённая 64-битная версия которой в 2019-ом году вышла под macOS с API Metal. По поводу Tomb Raider (2013), точных данных о используемом графическом API (Vulkan через MoltenVK или напрямую же задействуется API Metal) я не нашёл. Зато нашёл информацию касательно другой игры из серии Tomb Raider, а именно – Rise of the Tomb Raider, по ней разработчики, Fetal Interactive, написали в Твиттере, что они напрямую задействуют API Metal и что трансляция из Vulkan в Metal при помощи MoltenVK – не то, что им требуется. Позднее они написали, что они предоставляют наилучший возможный опыт для каждой платформы. Правда это или нет, в контексте стресс-теста для нас не важно, тут нам важно, чтобы видеоускоритель был максимально загружен, и чтобы стресс-тест не был ограничен по времени.

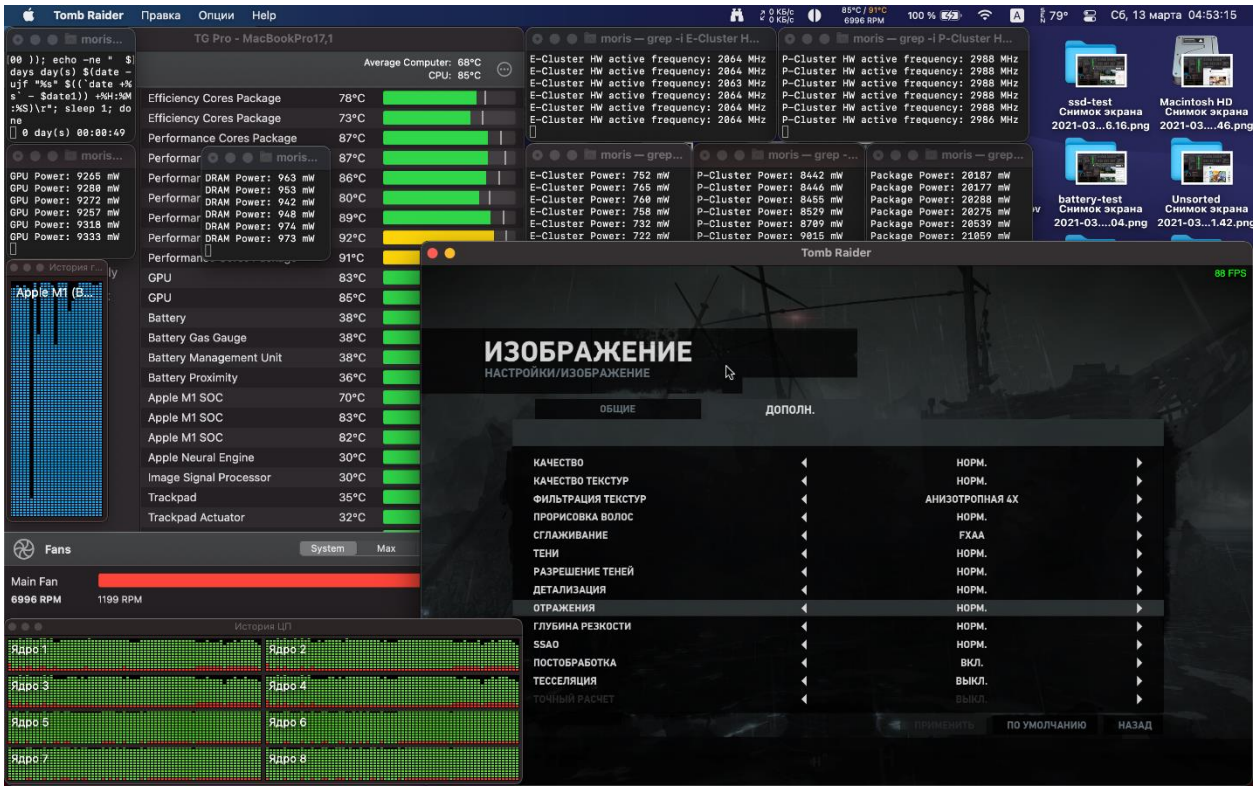
Тут как-раз то, что мы и хотим: в настройках игры циклически одна и та же сцена рендерится на заднем плане, и эта сцена, как оказалось, немало ресурсов потребляет. Мне удалось при пресете настроек графики «норм» таким образом загрузить видеоускоритель почти на 100% (потребление составило 9.5 Ватт). Процессор тут также был загружен просчётом физики к большому числу кадров в сцене, обрабатываемой видеоускорителем.



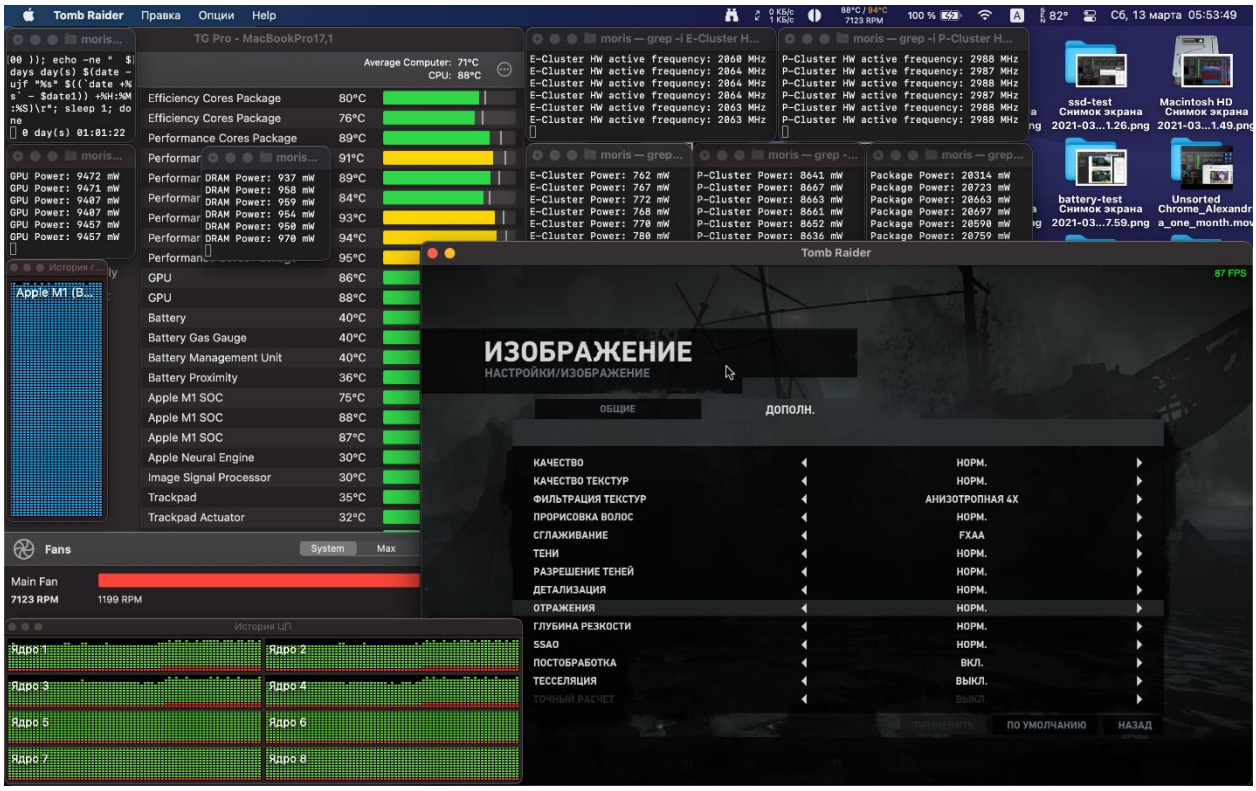
Скриншот 72. Macbook Pro 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре и команда yes x2.

Можно было бы загрузить видеокарту максимальными настройками графики и параллельно нагрузить процессор командой yes, как мы это делали ранее, но, если повысить настройки графики и дать нагрузку даже всего на 2 потока процессора командой yes, потребление энергии видеокартой упадёт уже до 7.5 Ватт. Причина, скорее всего, в том, что для отрисовки сцен в Tomb Raider процессор требует приоритет реального времени, чтобы успевать в срок обрабатывать физику для кадров, далее прорисовываемых видеоускорителем, потому лишний процесс yes, даже всего в 2 потока запущенный, вызывает снижение числа кадров и снижает нагрузку на видеокарту. В общем, в случае с Tomb Raider для того, чтобы полностью загрузить видеоускоритель и процессорные ядра, приходится идти другим путём: снижать настройки графики так, чтобы процессору приходилось чаще просчитывать физику для каждого кадра, далее прорисовываемого видеоускорителем.

Долго гадать не стали, сбросили секундомер и оставили вот так висеть настройки графики в игре.

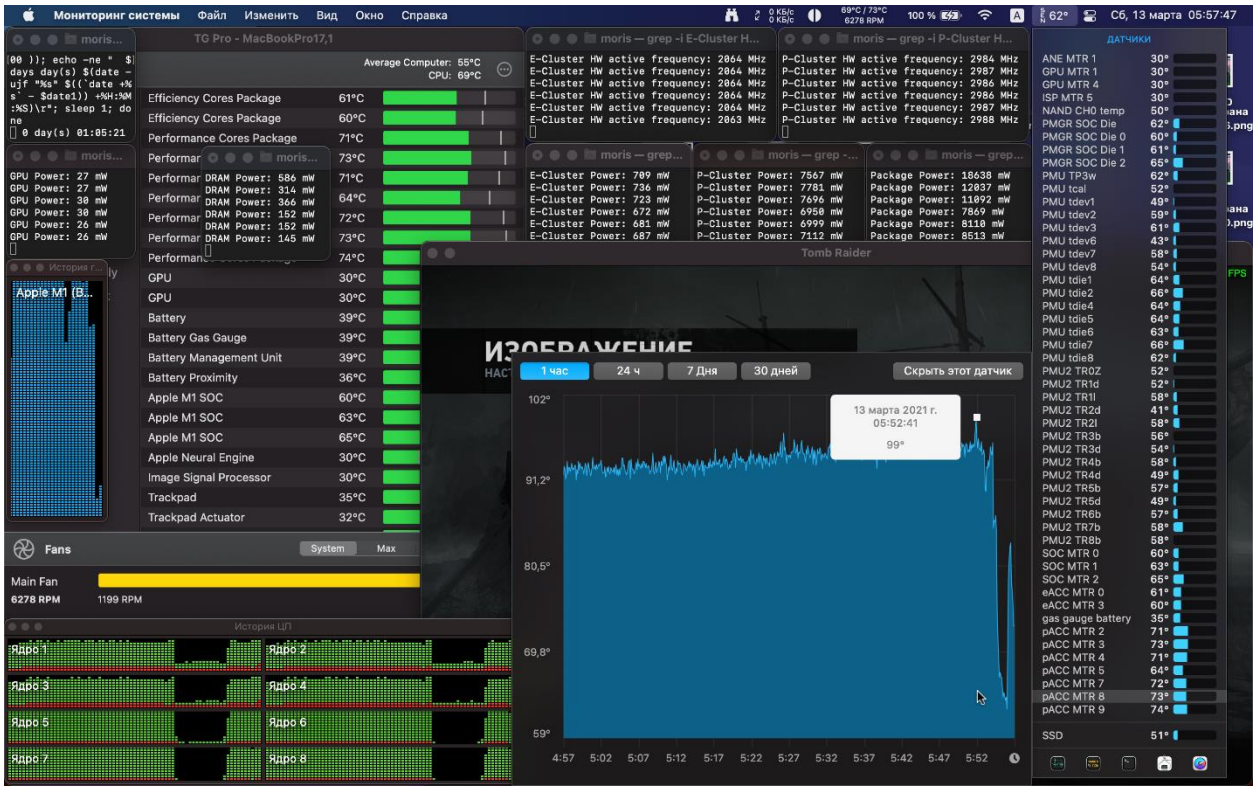


Скриншот 73. Macbook Pro 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 49 секунд.



Скриншот 74. Macbook Pro 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 1 час.

За час ничего не поменялось, т.е. троттлинга нет. Как был TDP 20 Ватт у SoC (за исключением DRAM), так и остался. Да и с температурами всё та же беда.



Скриншот 75. Macbook Pro 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 1.1 часа.

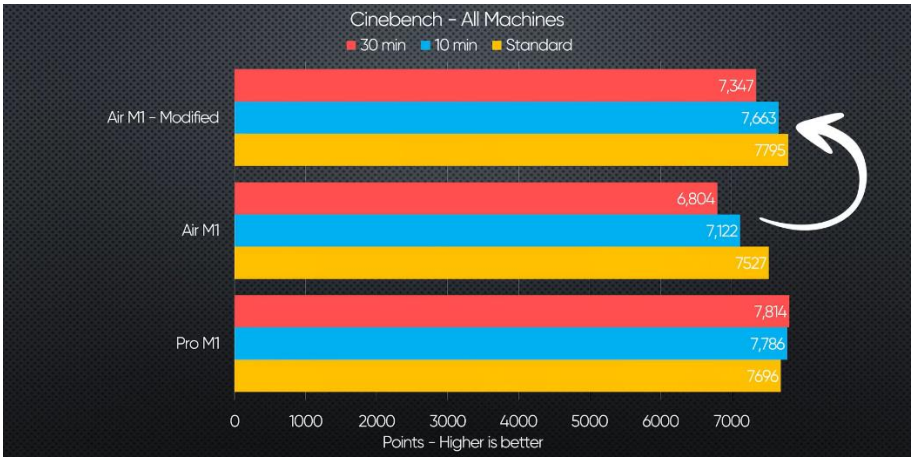
В общем, не поиграете Вы особо на Macbook с M1, либо поиграете, но недолго. То же самое касается и любого рабочего приложения. Кулеры и так крутятся на максимальных оборотах, тут править нечего. Можно лишь ограничить тепловыделение SoC... А, нет, нельзя, Apple такой возможности не дают. Apple, ну дали бы возможность выбрать сценарий работы «тише едешь, дальше будешь» и у меня бы вопросов не было: снизил бы производительность на 15% и не переживал бы за перегрев вообще.

Есть-что-нибудь хорошее? Да, при нагрузке на оба компонента (процессор и интегрированный графический ускоритель), предпочтение отдаётся графическому ускорителю, т.е. его производительность никак не урезается даже при наличии параллельной нагрузки на процессоре.

У меня складывается такое впечатление, что за разработку SoC Apple M1 отвечали эксперты в своей области, а за разработку охлаждения для него – новобранцы-дилетанты. Как можно было такое произведение инженерного искусства не раскрыть из-за косяка в охлаждении? Ну что за дела, Apple?

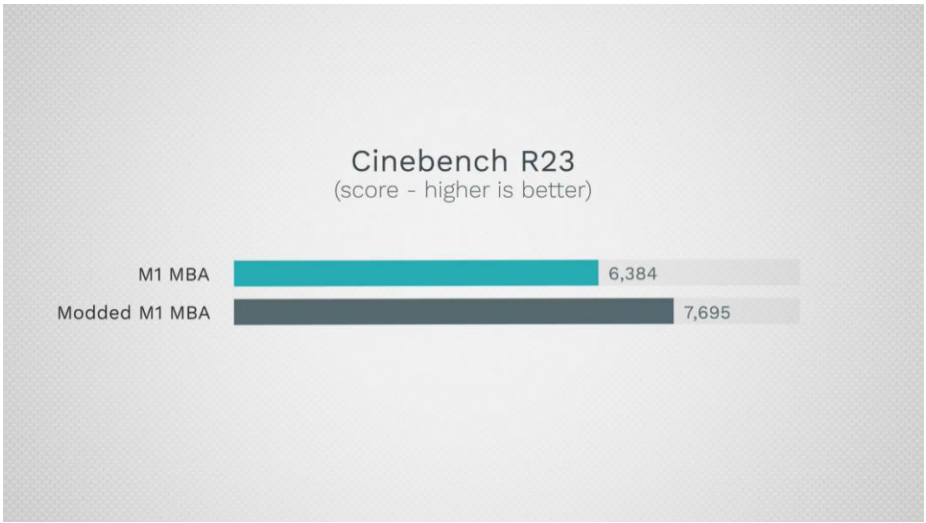
Более того, как оказалось, эффективность системы активного охлаждения в Macbook Pro настолько низка, что добавление термопрокладки в Macbook Air позволяет свести к минимуму разницу в их охлаждении.

Да, я не опечатался и Вам не показалось, простой пассивный отвод тепла с SoC на корпус ноутбука обычной термопрокладкой позволяет в задачах длительностью до 10 минут практически свести на нет разницу между Macbook Air и Macbook Pro по части охлаждения. Только не надо в какие-то крайности уходить и думать, что самая дешёвая термопрокладка тут сделает своё дело. Вы же, надеюсь, не станете на термопрокладке экономить 500 рублей, купив ноутбук за 100 тысяч рублей и более.



Видео 4. "I MODDED My M1 MacBook Air & Got 20% Performance Increase!" на канале Created Labs.

3 февраля на канале Linus Tech Tips вышел ролик «Fixing Apple's GOOD Engineering», в котором автор канала на 3:49 показал, как простая проклейка термопрокладки между теплораспределительной площадкой и корпусом ноутбука привела к снижению затрат времени на рендеринг в Blender на 8%, повышению производительности в Cinebench R23 на 7.5% при тестировании в течение 10 минут и на 8% при 30 минутах теста, что едва не равняет между собой Macbook Air и Pro с M1, когда у Air даже кулера нет.



Видео 5. "YOU NEED THIS on your M1 MacBook Air! (CPU Throttle FIX)" на канале Max Tech.

Следом 16 февраля вышел ролик «YOU NEED THIS on your M1 MacBook Air! (CPU Throttle FIX)», где на 6:54 автор канала Max Tech показал, что производительность Macbook Air при тестировании в Cinebench R23 в течение 10 минут с термопрокладкой выросла на 20,5%.

Далее на том же канале Max Tech вышло видео «Modded M1 MacBook Air vs M1 MacBook Pro: $30 Thermal FIX», где сравнили между собой Macbook Air с Apple M1 (версия на 7 ядер GPU вместо 8) с термопрокладкой, и Macbook Pro с Apple M1. Серьёзное отставание у Air наблюдалось только при экспорте 8K Raw видео с камеры Red в Final Cut Pro X на 10:05, там разница составила 16% в пользу Macbook Pro, но и то она вполне объяснима тем, что у Apple M1 в Macbook Pro имеется 8 графических ядер в сравнении с их Air с 7 ядрами GPU (у Pro нет версии с 7 графическими ядрами, тогда как у Air есть версии на 7 или 8 ядер GPU). По всем остальным тестам Max Tech у Air с термопрокладкой и Pro выходит паритет по производительности (где-то один чуть быстрее, где-то другой, но в целом паритет).

Macbook Pro на Apple M1 шумит, он греется до 99-101 градусов, он мерзко свистит своими кулерами под нагрузкой, а у Air из этих проблем только перегрев. Оба ноутбука греются, но Air с Apple M1 не шумит, т.к. там и шуметь то нечему без кулера, а по производительности он едва ли не такой же, как и Pro с Apple M1, при условии проклейки термопрокладки на Air.

Оба ноутбука нельзя назвать долговечными, т.к. под нагрузкой они будут сильно греться. Для того, чтобы процессор в ускоренном темпе не деградировал, его температура при работе не должна превышать 85 градусов (см. видео на канале Pro Hi-Tech с 07:52). Но, выбирая между греющимся ноутбуком, и аналогичным, только ещё и шумным, я предпочту первый. Не понимаю, зачем нужен Macbook Pro. Разве это чудо достойно звания «Pro»?

Выводы по троттлинг-тесту: охлаждение в Pro M1 хуже, чем в Pro 13 2019. Это едва не уровень RedmiBook 14 с Ryzen 3700U. Температуры выше, хотя тепла рассеять надо меньше. Более того, активное охлаждение тут настолько «никакое», что бесшумное пассивное в Air M1 с дополнительной термопрокладкой от него не сильно то и отстаёт. Да как так вышло то, Apple?

UPDATE: *Большое спасибо Роману Ястребу за его новый видеоролик, в котором он, разбирая видео Linus`a, ткнул в ошибку, которую и мы далее растиражировали в обзоре Macbook Pro на M1.*



Видео 6. Роман Ястреб: "MacBook Air M1 умрет с термопрокладкой! "

Роман Ястреб показал, почему нельзя на Air клеить термопрокладку: тепло с SoC (процессора, интегрированной графики, чипа безопасности, нейронного модуля и т.д.) таким образом будет отведено не только на корпус, а далее с корпуса вовне и рассеяно, но также тепло с корпуса будет таким образом отведено и на внутренние компоненты ноутбука, в т.ч. и на аккумулятор. Батарея ноутбука у Вас, таким образом, сильно нагреется и она очень быстро от этого деградирует, т.е. за короткий срок её износ будет и в самом деле весьма велик. В худшем случае она и вовсе вздуется.

В видео Роман проводил тест с Cinebench, который, в отличие от Blender, не грузит процессор в M1 на 100%. SoC Apple M1 в тесте Cinebench R23 потребляет всего 16 Ватт вместо 20, и выделяет оттого меньше тепла, и даже так температура аккумулятора у Романа достигала 45+ градусов, что для него крайне неблагоприятно. И это всего на 7 минут...

У меня к видео Романа только 2 поправки.

Первая поправка: даже если SSD нагреется до 70 градусов, это не критично. По-хорошему его температура не должна превышать 80-85 градусов, чтобы тот не деградировал, а вот критическая температура для SSD, как и для CPU, около 100 градусов.

Вторая поправка: 46 градусов – не критическая температура для акккумулятора. iStat Menus просто в качестве верхней планки/границы графика отображает максимальное значение, которое он зафиксировал, запрашивая у системы информацию о сенсорах каждые 4 секунды, а вот на самом графике информация обновляется лишь раз в 20-30 секунд. Так и получается, что на графике максимум – 45, а верхняя планка – 46. Ну и учтите, что значение для верхней планки округляется в большую сторону, если, например, была зафиксирована температура в 45.5 градусов. Не будет же верхней планкой в графике выставлено значение 45.5 градусов, а будет указано значение в 46 градусов, т.к. верхняя граница тут представлена всегда целым числом. Может ли аккумулятор работать при температуре 45-46 градусов? Да, может, но с течением времени при такой температуре он в лучшем случае быстро утратит ёмкость, в худшем – вздуется. Ну его...

В любом случае моя рекомендация взять Air вместо Pro и проклеить термопрокладку ошибочна. Я основывал свою рекомендацию на видео LinusTechTips и Max Tech, первого из которых и критикует Роман в видео.

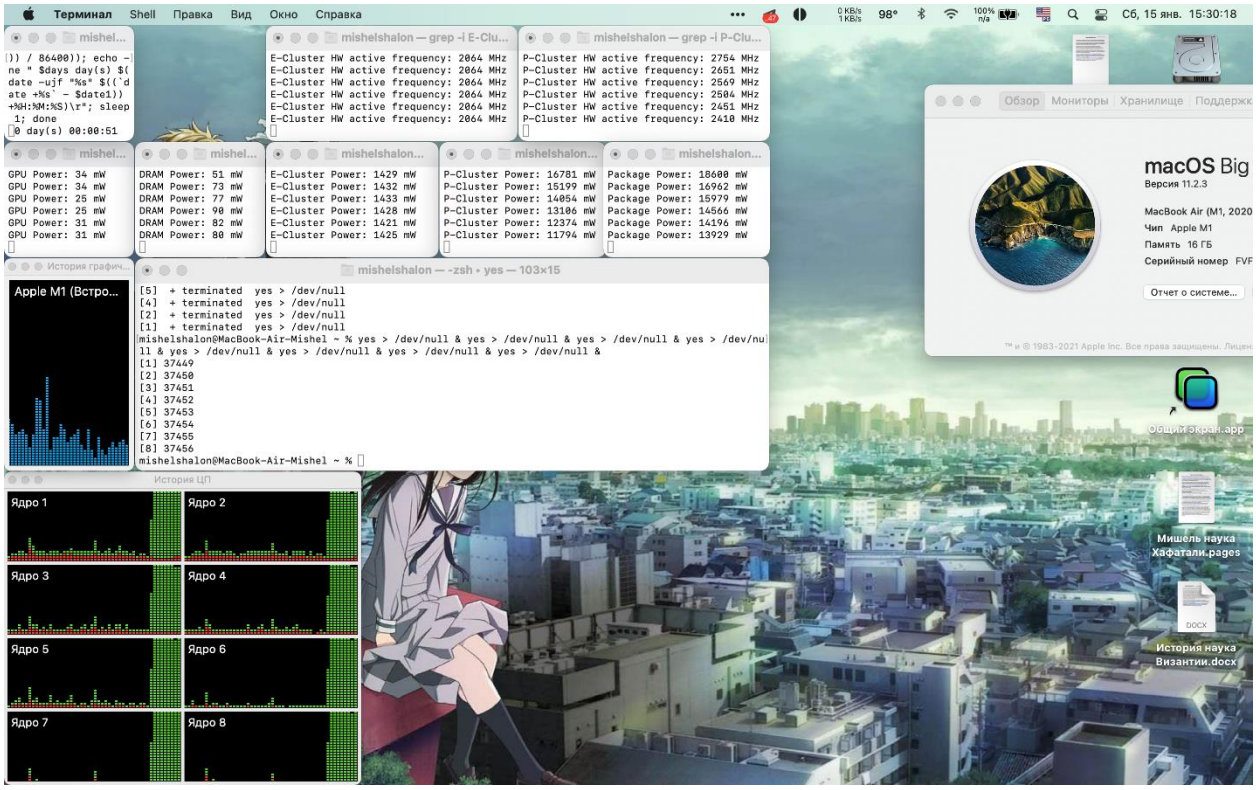
Но я всё ещё рекомендую брать Air вместо Pro, т.к. для полностью бесшумного ноутбука он безумно производительный. Если сравнить его результаты на Air со скриншота на 12:24, тот всего на 15%-34% (разница возрастает в течение теста первых 4 сцен) дольше рендерит сцены в Blender, но при этом никак не шумит, тогда как Pro в этих же тестах у меня свистел весьма ощутимо. Если прирост по производительности процессора в 30% при длительных нагрузках для Вас важнее отсутствия шума, берите Pro. Про проклейку термопрокладок на Air и думать забудьте.

Информацию мы со Стасом приняли, в отчёт об ошибках её включим.

У меня было 2 варианта, как поступить с этой информацией после того, как я её принял: или вырезать из своего обзора всякое упоминание термопрокладок, или просто дополнить обзор дополнительной информацией. Если бы я выбрал первое, Вы бы никак из этого обзора не узнали, что ни в коем ради не нужно клеить термопрокладку на Air, потому я выбрал второе. Надеюсь, мысль я донёс, и Вы не совершите эту ошибку.

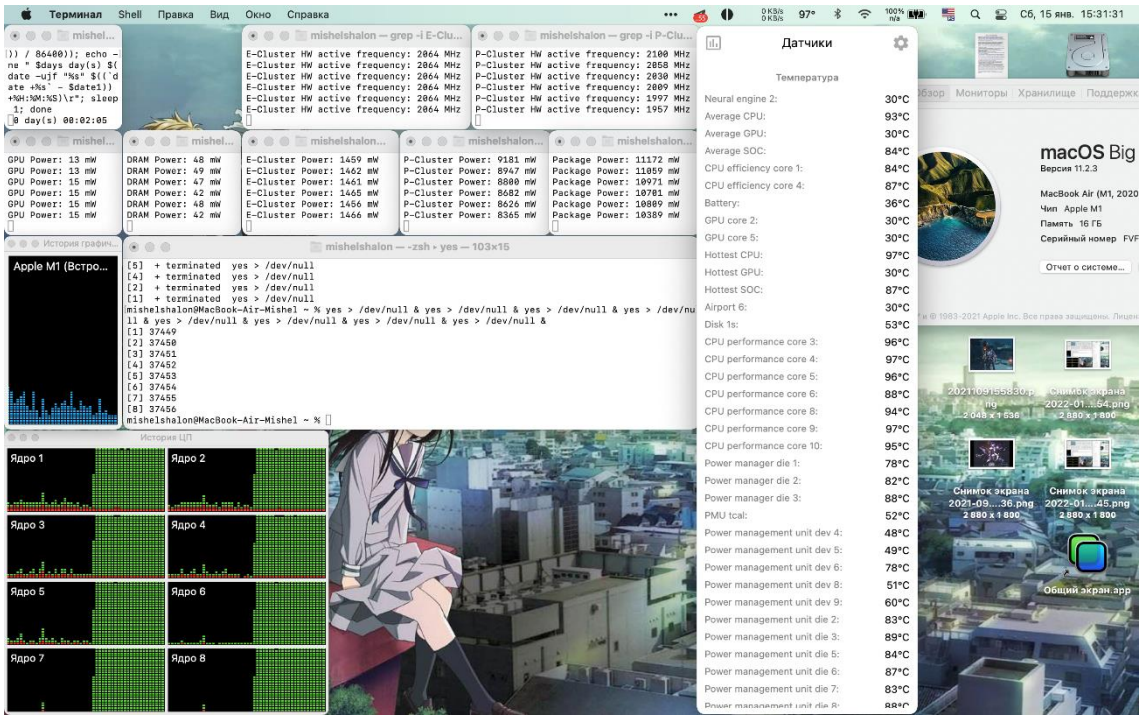
UPDATE 2: *спустя полгода после написания обзора на Macbook Pro на базе Apple M1, в сентябре 2021-го мы вместе с моим братом собрали деньжат и [подарили моей сестре Macbook Air на Apple M1](#) (брали в версии 16/512). И спустя ещё несколько месяцев, в январе 2022-го, т.е. спустя почти год после публикации статьи, я одолжил у сестры её Macbook на время, и прогнал ещё несколько тестов. И тут вскрылся нюанс: троттлинг у Macbook Air на самом деле просто огромный, и не важно, грузим ли мы CPU, или же грузим GPU.*

Для отслеживания температуры в этот раз я воспользовался [Stats](#). Разницы по показателям с iStat Menus и другими утилитами я не заметил. Они все запрашивают информацию у системы по одним и тем же API. [Stats](#) просто, в отличие от других, бесплатный и с [открытым исходным кодом](#).



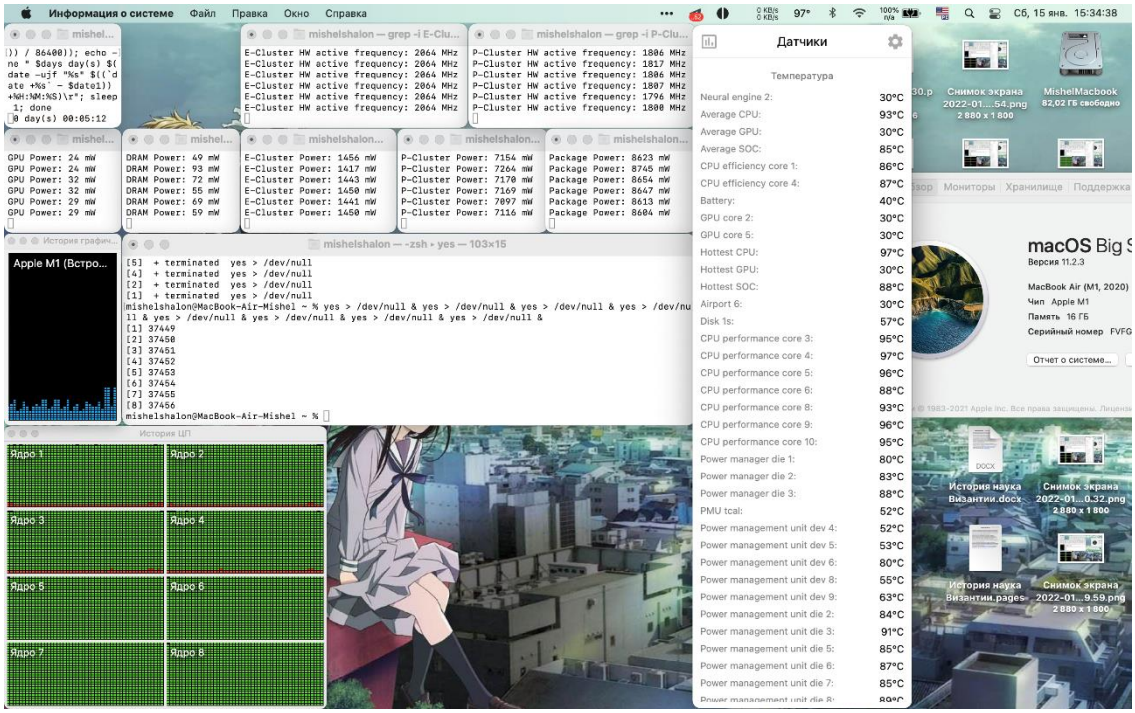
Скриншот 76. Macbook Air 13 2020 (Apple M1). Стресс-тест командой yes в 8 потоков. Спустя 1 минуту.

Спустя всего 1 минуту процессор в Air уже вовсю бьётся в температурный порог в 100 градусов, и скидывает тепловыделение процессора. Как он это делает? Снижает тактовую частоту его производительных ядер. Снизив частоту с 3 ГГц в начале (она тут на старте такая же, как у Pro) до 2.4 ГГц (у Pro на постоянке были 2.8 ГГц), ему удаётся снизить тепловыделение с 22 Ватт до 14. Но это ещё не всё.



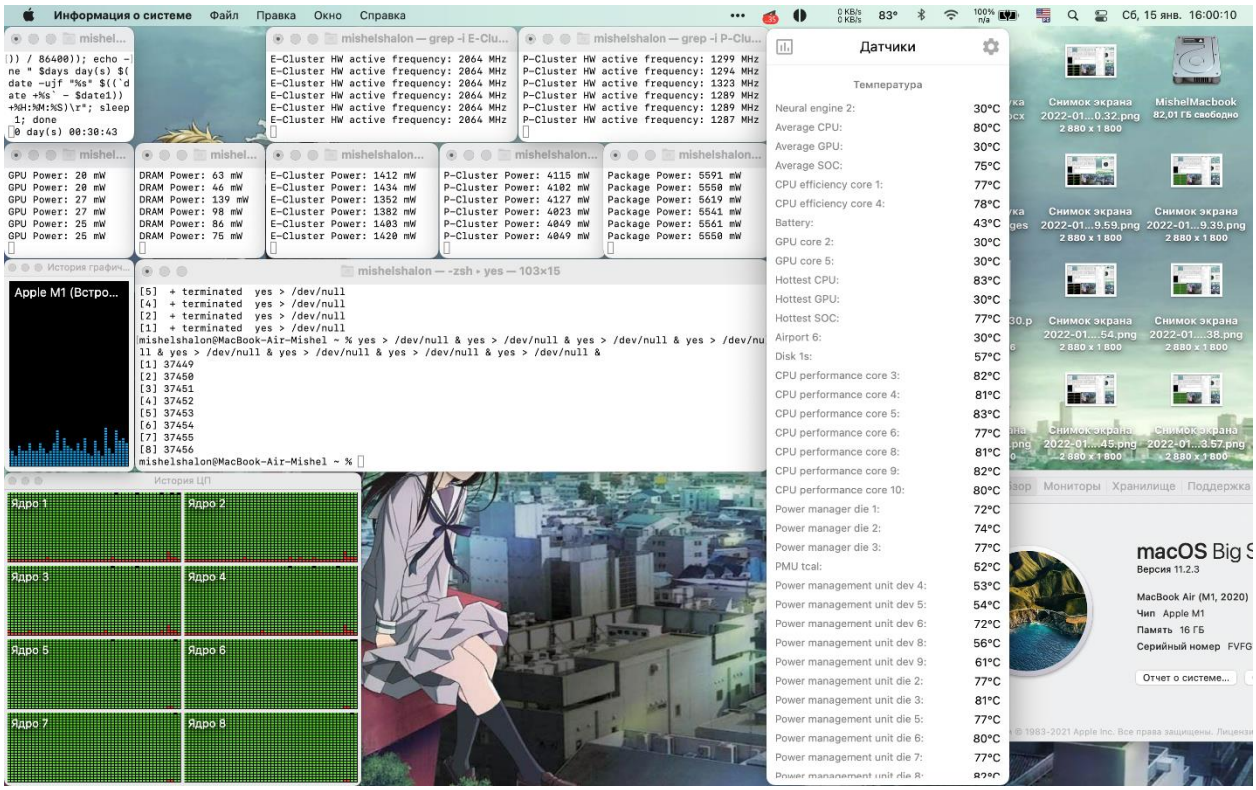
Скриншот 77. Macbook Air 13 2020 (Apple M1). Стресс-тест командой yes в 8 потоков. Спустя 2 минуты.

Спустя 2 минуты тепловыделение падает уже до 10.5 Ватт. За счёт чего это происходит? За счёт урезание частоты производительных ядер аж до 2 ГГц. В начале 3 ГГц, а спустя 2 минуты уже 2. Но и это не конец.



Скриншот 78. Macbook Air 13 2020 (Apple M1). Стресс-тест командой yes в 8 потоков. Спустя 5 минут.

Спустя 5 минут тепловыделение у Air падает до 8.5 Ватт. В плату за это частота процессора урезается до 1.8 ГГц. Но, казалось бы, всё, теперь всё устаканилось. Да, процессор жарит постоянно до 100 градусов, но дальше ведь частота ещё сильнее не упадёт, верно?



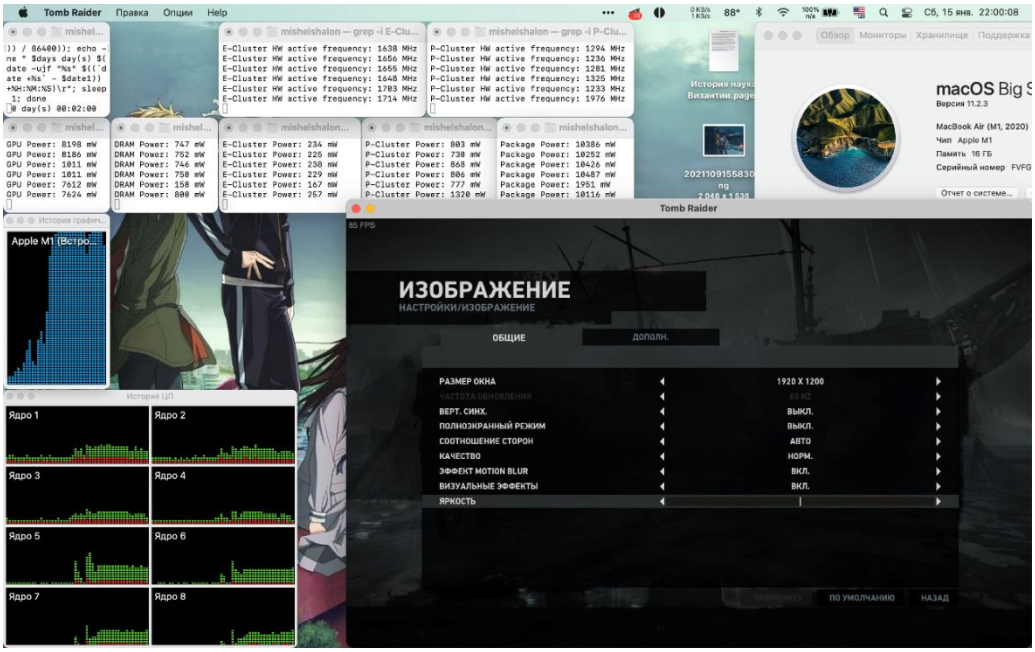
Скриншот 79. Macbook Air 13 2020 (Apple M1). Стресс-тест командой yes в 8 потоков. Спустя 30 минут.

Не знаю, по какой причине, но Air и Pro в троттлинг-тесте ведут себя по-разному. Один (Pro) постоянно бьётся в температурный порог в 100 градусов, а другой, как оказалось, спустя время опускает этот порог. Спустя пол часа у Air температура не превышала уже 85 градусов. С одной стороны, это круто, ведь так выходит, что он дольше прослужит, т.к. реже будет перегреваться. А с другой: а как температура снизилась то? Кулеры, что ли, возникли из неоткуда у Air? Нет, он ещё сильнее скинул частоту, чтобы сбавить тепловыделение процессора. Частота упала до 1.3 ГГц, и так ему удалось сбавить тепловыделение до 5.5 Ватт. Спустя время она обратно уже не поднималась, т.е. спустя час картина всё та же.

Понимаете разницу? У Pro процессор постоянно вжаривал на полную, на максимум который позволяла система охлаждения, сколь бы фиговой она ни была. А Air сперва урезает тепловыделение почти в 2.5 раза в сравнении с Pro, а затем и вовсе решает, что надо бы быть холоднее, и срезает его почти в 4 раза за счёт снижения частоты более чем в 2 раза. В 2, Карл!

Т.е. при длительных нагрузках в задачах, где частота процессора будет одним из наиболее важных критериев для быстрого решения задачи, Air на M1 будет уступать Pro на M1 в !!! 2 !!!, с@ка, раза! Ахренеть.

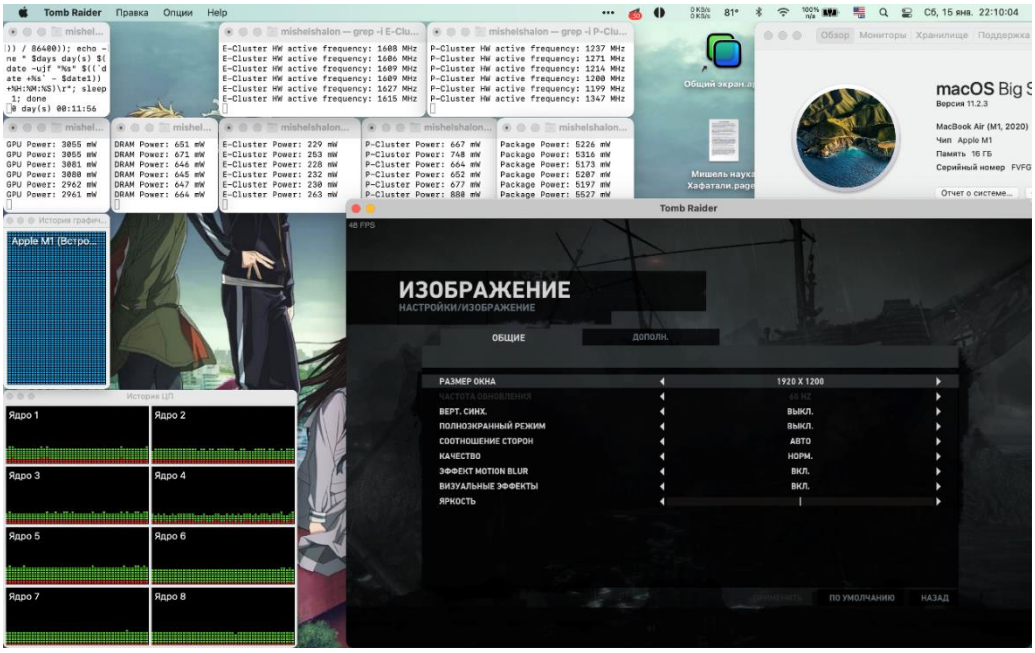
Хорошо. Может, это всё не столь важно. Ведь маки многие берут для работы с графикой. Может, 5.5 Ватт загруженному GPU вполне хватит?



Скриншот 80. Macbook Air 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 2 минуты.

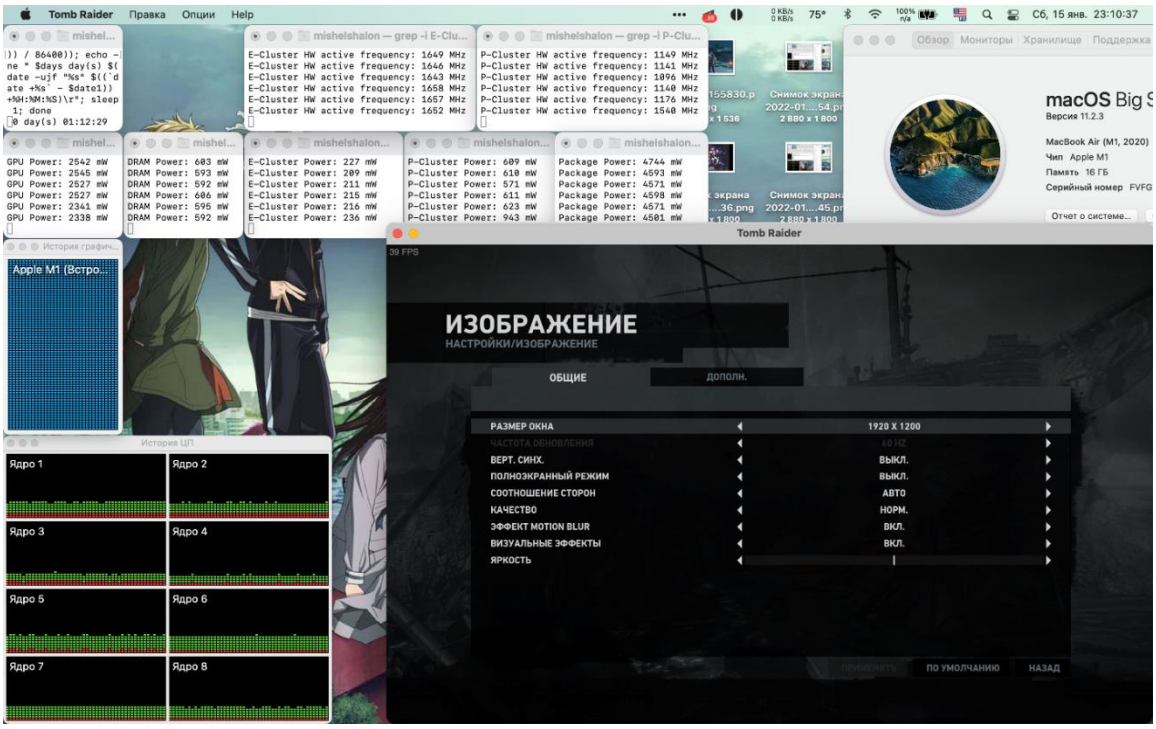
Я запустил на Air ту же панель настроек графики в Tomb Raider, что запускал и на Macbook Pro с M1. В первые 2 минуты FPS был около 85. Примерно та же картина, что и на Pro с M1. Троттлинг ещё не заметен.

И не имеет значения, 7 ядер у GPU или 8. В макбуке сестры 7 ядер у GPU, но, пока троттлинга нет, производительность почти как у Pro с 8.



Скриншот 81. Macbook Air 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 12 минут.

Спустя 12 минут FPS упал до 48. Температура снизилась до 81 градуса ввиду снижения тепловыделения SoC до 5.0-5.5 Ватт.



Скриншот 82. Macbook Air 13 2020 (Apple M1). Tomb Raider (2013). Панель настроек графики в игре. Спустя 1 час.

Спустя час FPS и того упал до 40. Температура также снизилась до 75 градусов, т.к. тепловыделение процессора упало до 4.5 Ватт. Т.е. Air в сравнении с Pro с M1 в задачах, где требуется высокая производительность графики, также оказывается примерно в 2 раза медленнее.

А я ещё думал, почему у сестры на Macbook Air спустя пол часа игры в Genshin Impact притормаживать начинает, словно FPS раза в 2 падает...

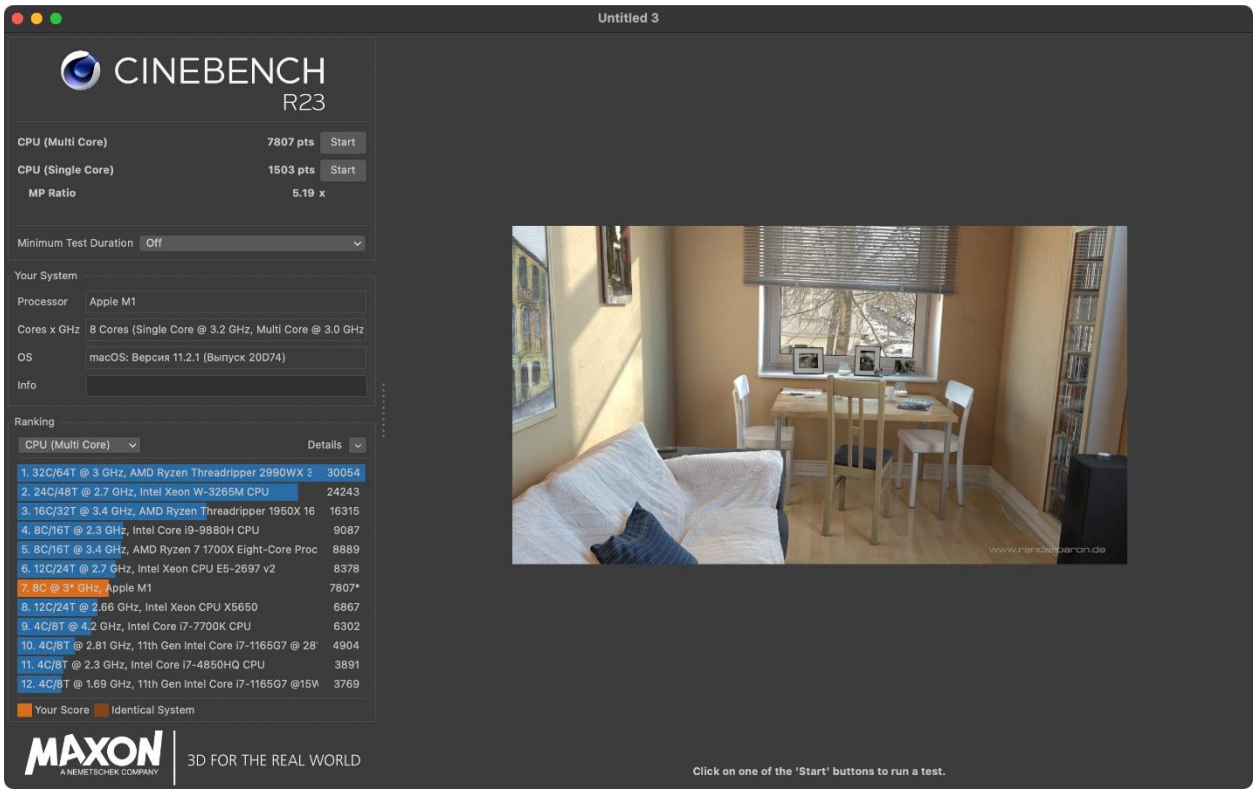
Вывод тут какой? Air не предназначен для длительных нагрузок. Если его грузить 10 минут и более, он будет сильно медленнее, чем Pro. Спустя пол часа разница доходит до 2 раз. И так что по части нагрузки на CPU, что по части нагрузки на GPU. Air, может быть, мог бы быть и шустрее, но Apple на Air с M1, в отличие от Pro, спустя время старается держать более низкую температуру. Поэтому, если хотите взять мак для серьёзной работы, доплатите уже 20 тыс. и возьмите себе Pro. Нам с моим братом этот Macbook Air (16/512) обошёлся примерно в 120 тыс. руб., и грустно то, что он теряет половину своей производительности спустя пол часа... Лучше бы доплатили тогда и купили бы Pro. Но тут уж извините, куда деваться.

На случай, если вдруг выяснится какая-то ещё интересная информация, можете подписаться на мой Telegram-канал по технологиям, я там первым же делом об этом отпишу, как сделал и в данном случае.

5. Тесты производительности процессора.

Сперва мы проведём те же тесты, которые Вы и так можете найти в интернете, чтобы не столько протестировать Macbook Pro 13 на Apple M1, сколько проверить ещё производительность Xiaomi Mi Notebook Pro GTX с i7 8550U и, по результатам в сети, Вы сможете сами сопоставить его с Macbook Pro 13 2018 с 4 портами Thunderbolt 3 и убедиться, что, по сути, если мы затрагиваем только производительность центрального процессора (без учёта графики), он по производительности от MBP 13 2018 с 4x TB3 не отличается (либо разница на грани погрешности). Также мы провели часть тестов и на двух ноутбуках Macbook Pro 13 с 4 портами Thunderbolt 3: это модели 2018 года Кирилла Варшамова (kavars) с чипом i5 8259U и 2020 года Владимира Баруса с чипом i5 1038NG7). Данные, собранные в первых тестах, позволят нам понять, можно ли говоря о превосходстве над моим Xiaomi, говорить и о превосходстве над MBP 13 Core i5 с 4 портами Thunderbolt 3.

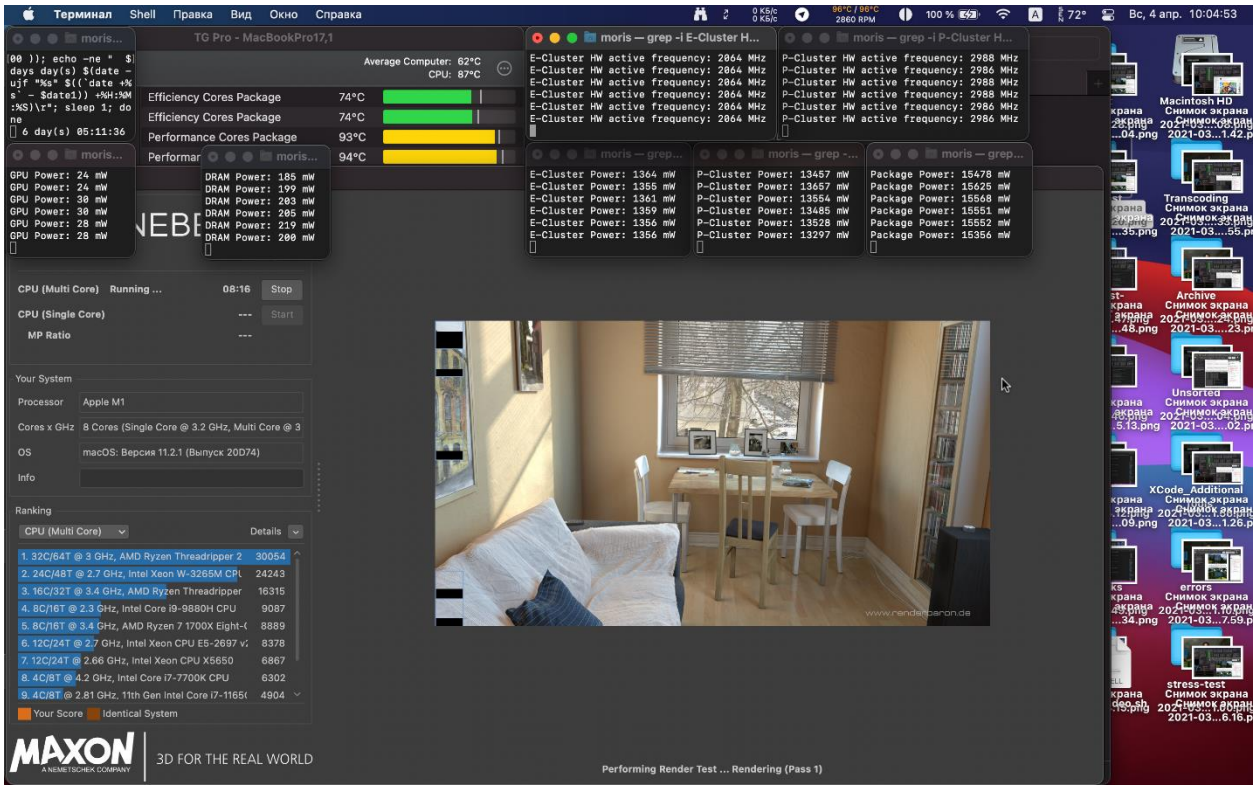
5.1. Cinebench R23.



Скриншот 83. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Cinebench R23. Нативно (ARM mode).

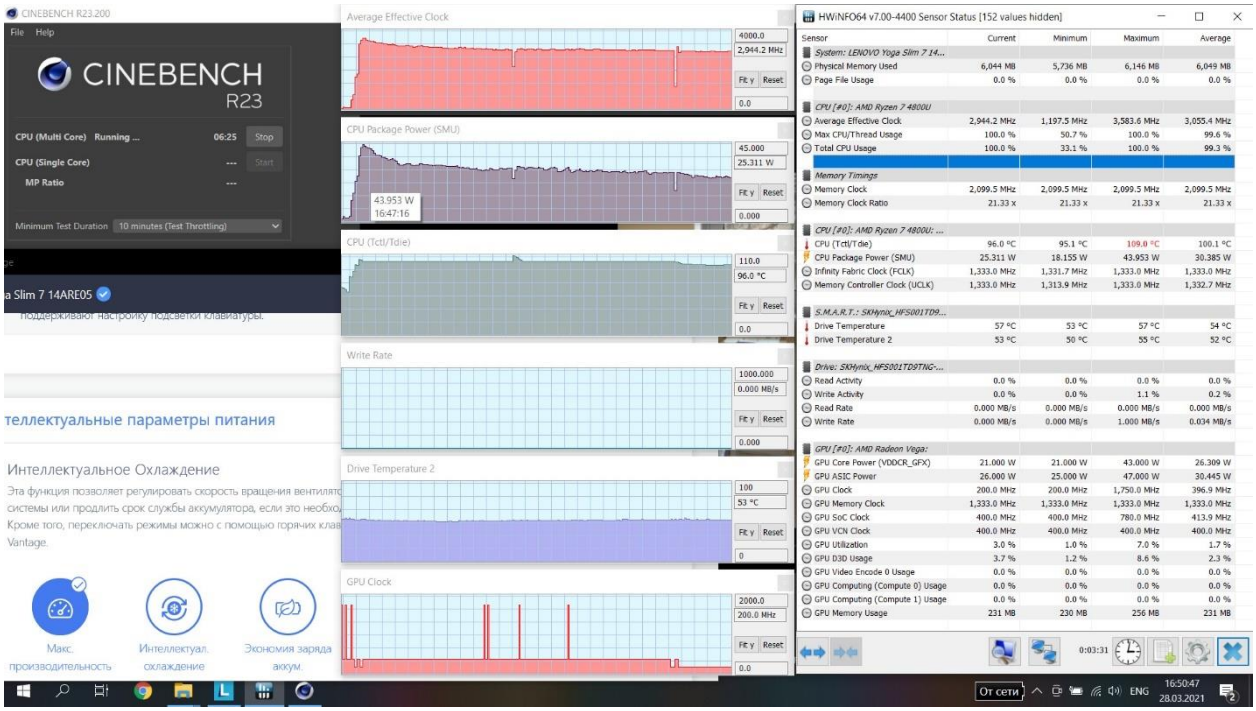
Macbook Pro 13 2020 с чипом Apple M1 при TDP 20 Ватт 7807 баллов, что на 16.2% ниже, чем Lenovo Yoga Slim 7 с Ryzen 7 4800U на 25 Ватт.

Но тут требуется важное уточнение.



Скриншот 84. Macbook Pro 13 2020 (Apple M1, 20W). Мониторинг CPU в Cinebench R23. Нативно (ARM mode).

Дело в том, что в тесте Cinebench R20 новый Macbook Pro на Apple M1 потреблял не более 16 Ватт энергии и кулеры даже толком не раскрутились. Более того скажу: я потребления выше 22 Ватт в принципе ни разу не видел у Macbook Pro на M1.



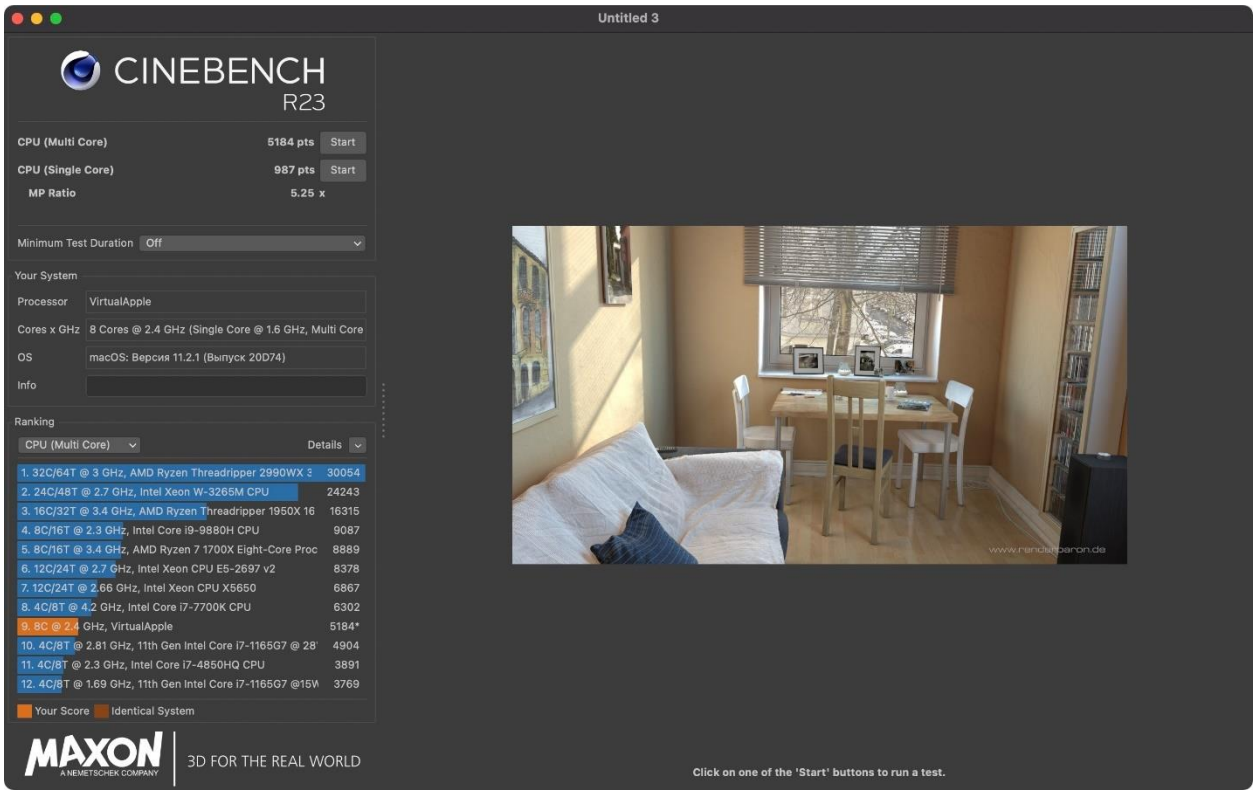
Скриншот 85. Lenovo Yoga Slim 7 (Ryzen 7 4800U, 25W). Мониторинг CPU в HWinfo при тестировании Cinebench R23.

В то же время, Lenovo Yoga Slim 7 «вжарил» на старте на все 43 Ватта.

За первую минуту после начала теста у Lenovo плавно опустилось тепловыделение до 30 Ватт, затем за 2 минуты – ещё до 25 Ватт. Вот только если не устанавливать длительность теста в 10 или 30 минут, то тест на Lenovo завершится всего за 2-3 минуты, и его среднее потребление энергии за это время превысит 30 Ватт. А Macbook Pro на Apple M1 потреблял 15-16 Ватт. Т.е. при потреблении энергии в 2 раза ниже, Mac на M1 отстал всего на 16.2%, если за 100% принимать результат Lenovo Yoga Slim с Ryzen 7 4800U.

Энергоэффективность – вот он конёк новых Macbook на Apple M1. Во многом тут сказывается разница между декодированием команд при помощи микрокода из сложных в мелкие микро-операции, и декодировании при помощи аппаратных блоков управления. Тут решает компилятор, который сразу под RISC нативный ПО собирает.

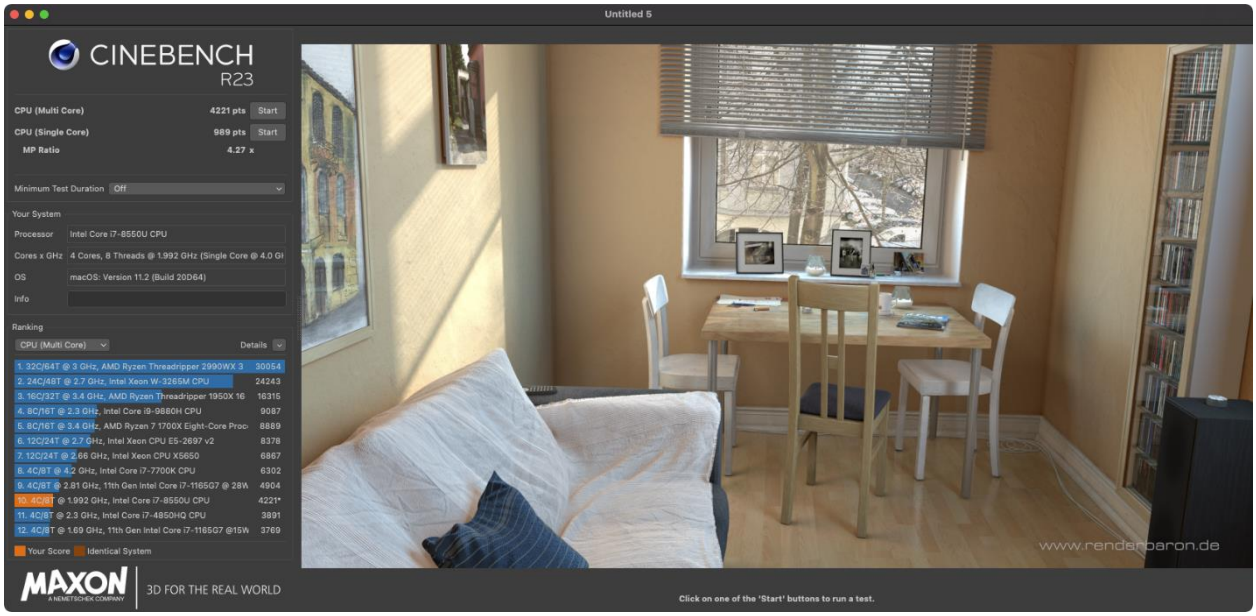
А если смотреть на однопоточную производительность, у него и конкурентов то практически нет среди решений для ноутбуков. У него производительность на 1 поток даже чуть выше, чем у Asus ROG Strix G15 с Ryzen 9 5900HX на 7 нм Zen 3, который совсем недавно был у нас на тесте.



Скриншот 86. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Cinebench R23. Rosetta 2 (Intel mode).

Если запустить Cinebench R23 через Rosetta 2, тем самым транслировав часть кода x86 в универсальном исполняемом файле приложения (который

содержит машинный код и под x86, и под ARM), производительность снизится в 1.5 раза (-33,6% с Rosetta 2 или прирост +50,6% при отключении Rosetta 2). Это колоссальная разница, M1 с Rosetta 2 теперь на 44.4% отстаёт от Lenovo Yoga Slim 7 с Ryzen 7 4800U, т.е. разница примерно равна разнице в энергопотреблении. Зато M1 всё ещё опережает Core i7 1165G7 на 15 Ватт в многопоточном режиме на 29.8%, т.е. M1 всё ещё эффективнее, чем CPU от Intel, а с AMD паритет. В однопоточном режиме M1 сравнялся с ... Xiaomi???



Скриншот 87. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Результат теста CPU в Cinebench R23.

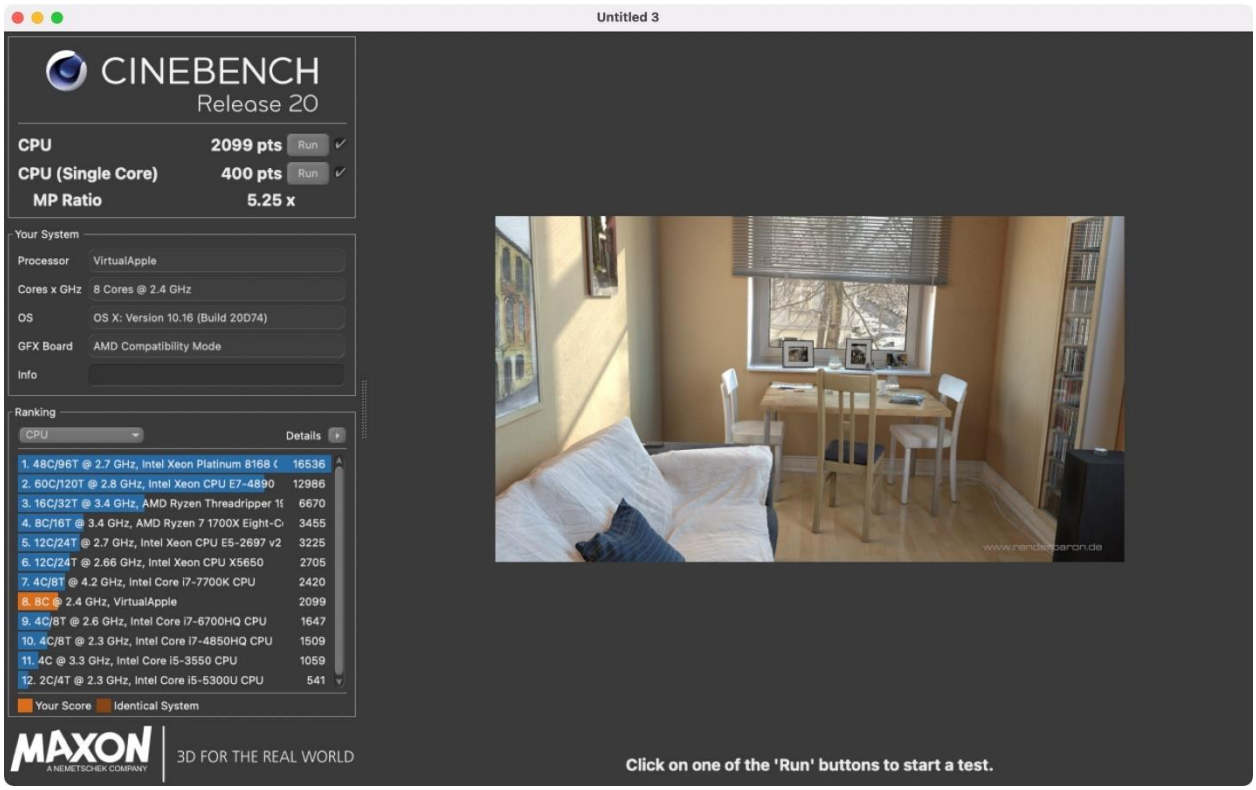
Rosetta 2 нельзя использовать для трансляции кода ARM приложений в x86, так что в случае с Xiaomi у нас только один режим работы: нативный. По производительности процессора в многопоточном режиме Xiaomi с i7 8550U при TDP в 25 Ватт немного опережает i7 1165G7 в ноутбуке Asus Zenbook Flip S, а в однопоточном режиме он равен Macbook Pro 13 2020 с чипом Apple M1 при запуске Cinebench R23 через Rosetta 2. Данные Macbook Pro 13 2018 с 4 портами Thunderbolt 3 (Core i5 8259U) вы без труда сможете найти в интернете, и там Вы увидите, что разница между ним и Xiaomi на грани погрешности, так и тут разница между i7 8550U в ноутбуке Xiaomi и Apple M1 в Macbook Pro - на грани погрешности в однопоточном режиме при работе последнего через Rosetta 2. В многопоточном режиме Macbook Pro ещё даст им фору. Так, M1 опережает i7 8550U в многопотоке на 22,8%.

5.2. Cinebench R20.

Компания-разработчик Cinebench (Maxon) удалила со своего оф. сайта версию Cinebench R20, которая ещё не поддерживала нативно компьютеры с Apple Silicon, но я сумел найти старую версию на сайте techspot.

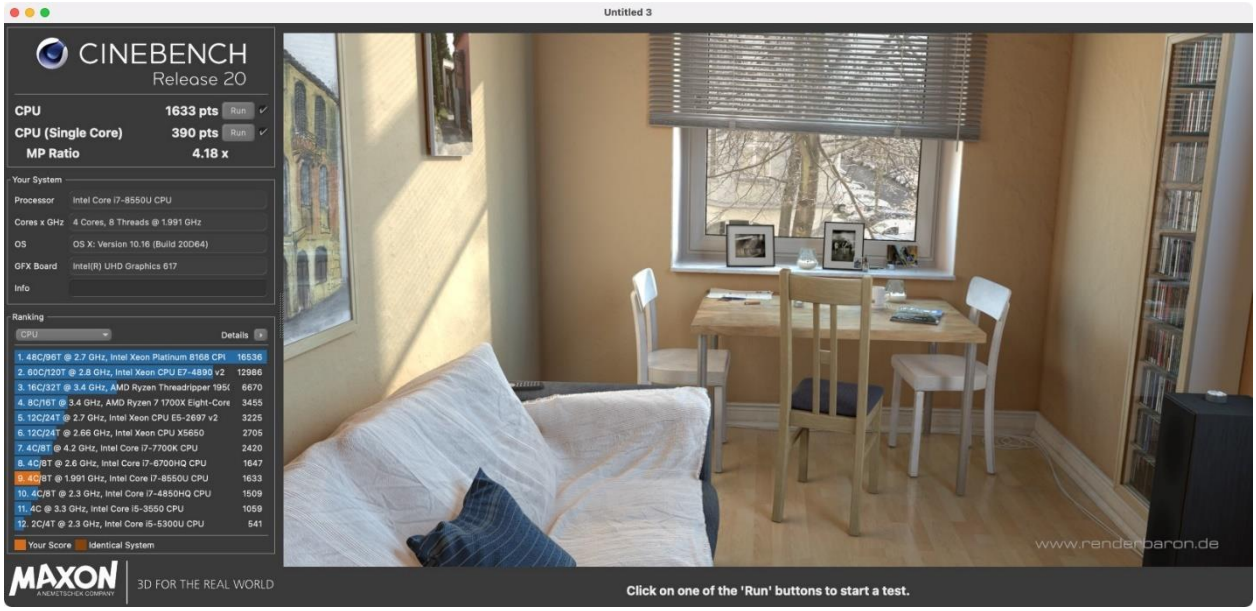
Зачем проводить старый тест? Просто чтобы выявить, будет ли разница в производительности между M1 и Xiaomi такой же, если мы запустим через Rosetta 2 приложение, которое в принципе не имеет версии под ARM.

В целом не должно быть разницы, транслируете ли вы через Rosetta 2 всё приложение в случае с приложением только под x86 или же часть машинного кода для x86 в случае с универсальным приложением, но мы в любом случае это проверим, т.к., мало ли, вдруг мы не правы.



Скриншот 88. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Cinebench R20. Rosetta 2 (x86 only).

С Rosetta 2 в Cinebench R20 Macbook Pro на M1 по производительности обошёл Core i7 1165G7 в Asus Zenbook Flip S, но уже не на 29.8%, а всего на 19.4%. Выходит, всё же, трансляция приложений целиком даётся тяжелее? Нет, отставание от Ryzen 7 4800U на 30-32 Ватта в среднем в первые 2-3 минуты составило 42.5%, что примерно сопоставимо с тем, что было в Cinebench R23. Сравним ещё с Xiaomi и будет понятнее.



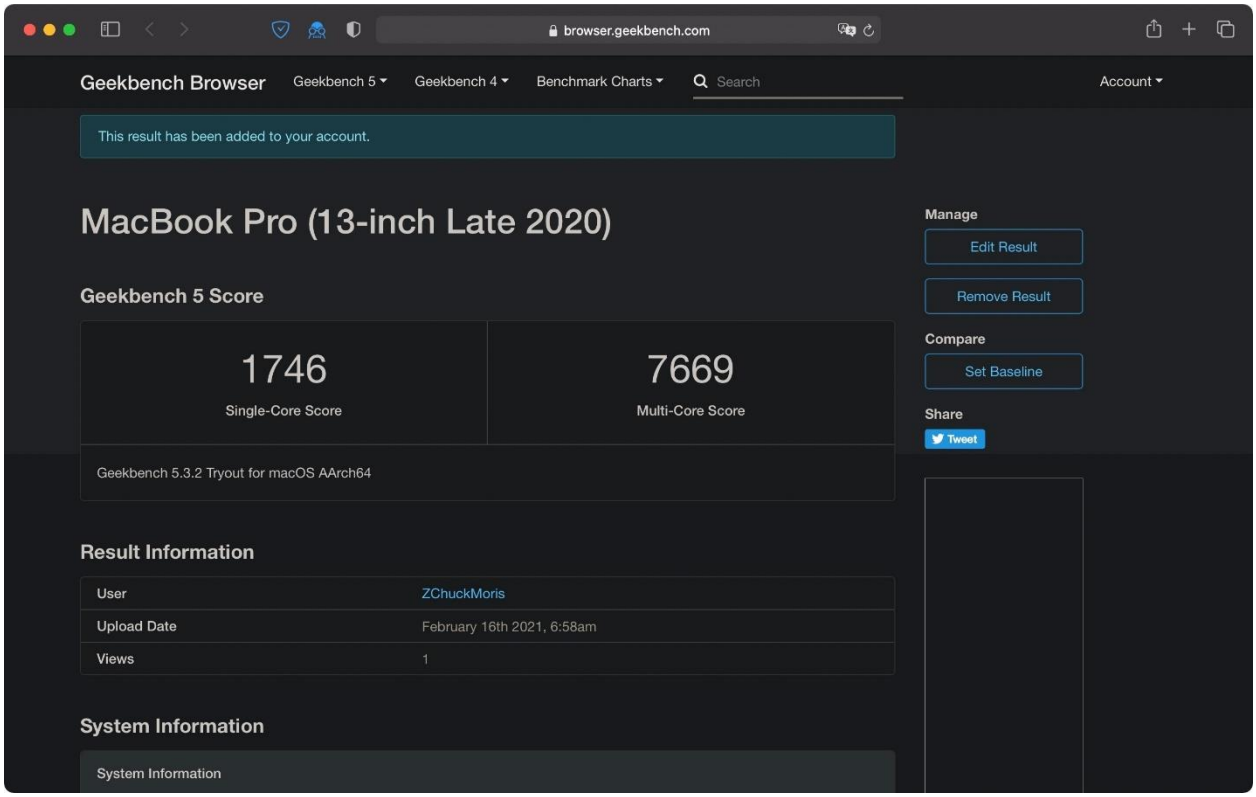
Скриншот 89. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Результат теста CPU в Cinebench R20.

В случае с Xiaomi, наоборот, разница возросла в пользу Macbook Pro с M1: была 22.8%, а стала 28.5%. Полагаю, что в Cinebench R23 просто были изменены алгоритмы рендеринга изображений, и именно эта разница и даёт небольшие отклонения в ту или иную сторону при сравнении результатов аппаратов с Cinebench R20 и Cinebench R23. И, как видите, эти изменения затрагивают не все процессоры, т.к. разница осталась примерно той же между Apple M1 и Ryzen 7 4800U, а также между Apple M1 и Core i7 8550U.

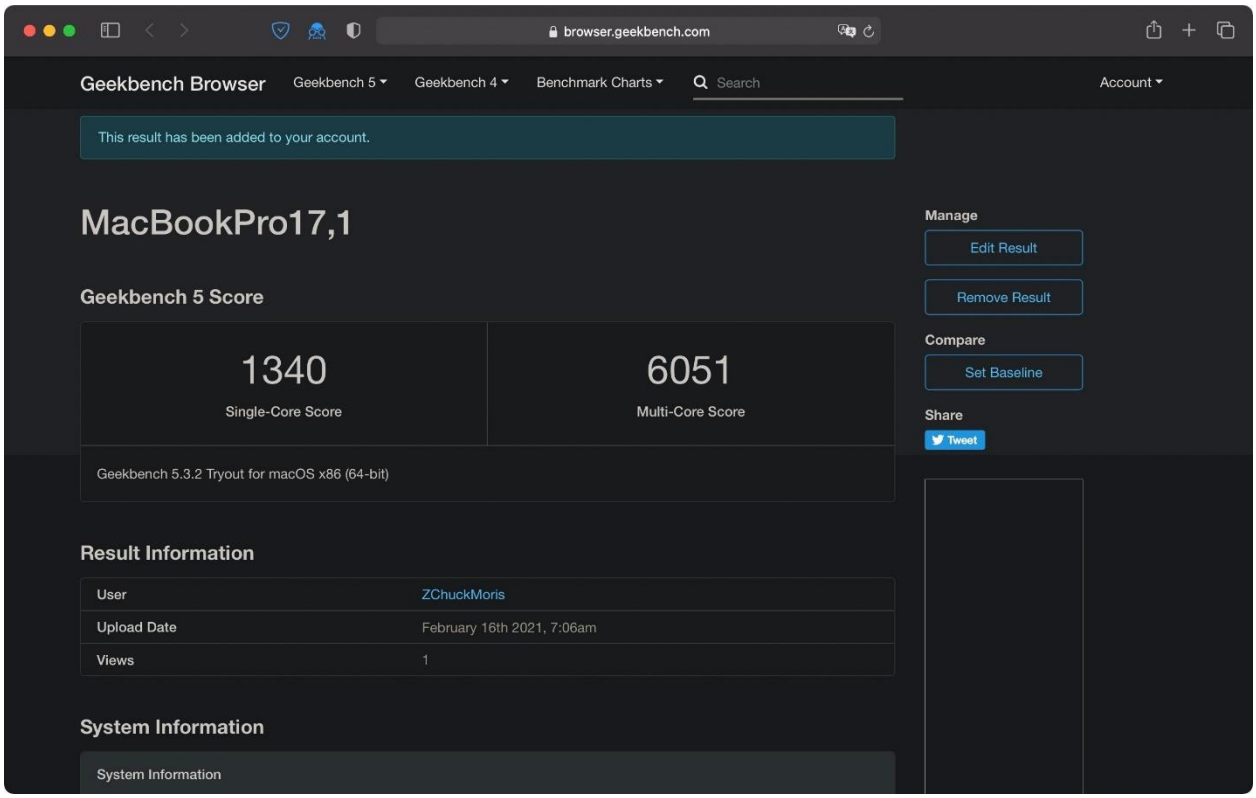
Пока что Macbook с 2 портами Thunderbolt 3 и чипом Apple M1 с Rosetta 2 выходит более предпочтительным вариантом, чем Macbook Pro 13 2018 или 2020 с 4 портами Thunderbolt 3 (опять же, их результаты легко найдутся по этому тесту в сети). Но если сравнивать с Windows ноутбуками с процессорами AMD Ryzen 7 4800U на 8/16 ядер потоков, то разница в производительности выходит более или менее сопоставимой с разницей в энергопотреблении. Но это только что касается производительности в многопоточном режиме, когда задействуются все 8 ядер у Apple M1, включая 4 энергоэффективных ядра и 4 высокопроизводительных ядра, и все 16 потоков у Ryzen 7 4800U. Однако же вспомните, что энергоэффективные ядра в Apple M1 потребляют не более 2 Ватт под максимальной нагрузкой, их влияние на результаты невелико. Т.е. 5 условных производительных ядер в Apple M1 всего в 2 раза слабее 8 ядер и 16 потоков в Ryzen 7 4800U. Boy.

5.3. Geekbench 5.

Тут мы также провели 2 теста на Macbook Pro с M1: с Rosetta 2 и без.

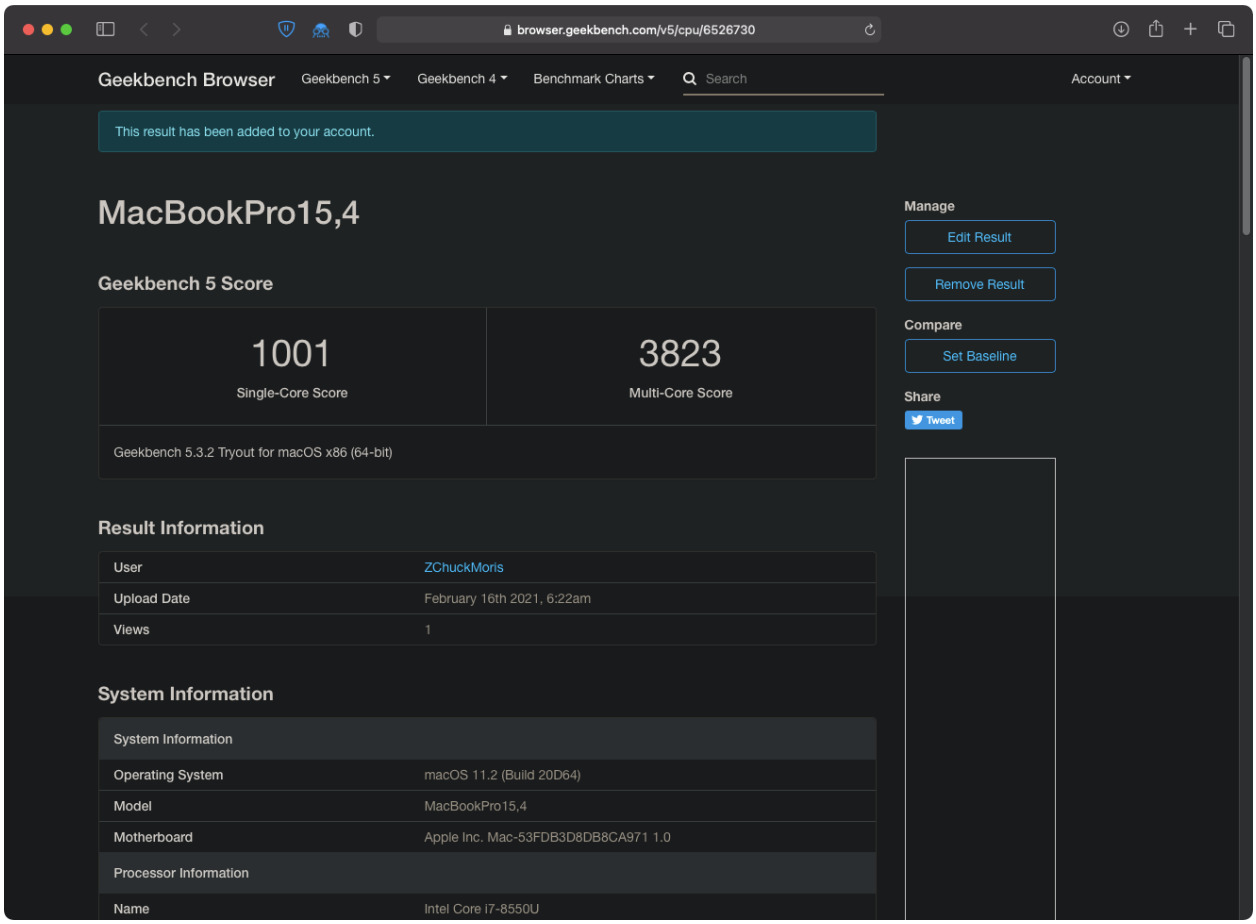


Скриншот 90. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Geekbench 5. Нативно (ARM mode).



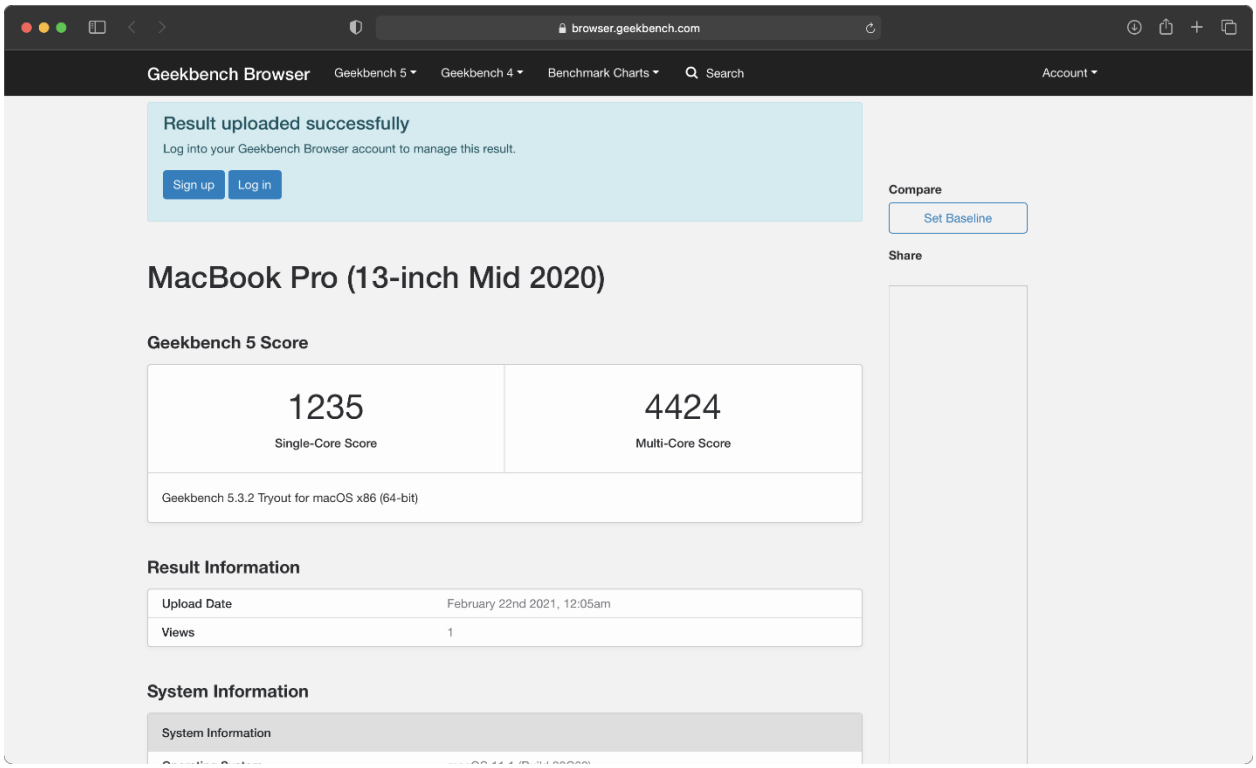
Скриншот 91. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Geekbench 5. Rosetta 2 (Intel mode).

В Geekbench 5 разница между запуском с Rosetta 2 и без составила 1.27 раза (прирост 26,74% без Rosetta 2 или потеря 21.1% при её использовании).



Скриншот 92. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Результат теста CPU в Geekbench 5.

Macbook Pro с M1 на 58% опережает Xiaomi в Multi Core с Rosetta 2 и на 101% (в 2 раза) опережает без неё, а в Single Core – на 34% с и 74% без.

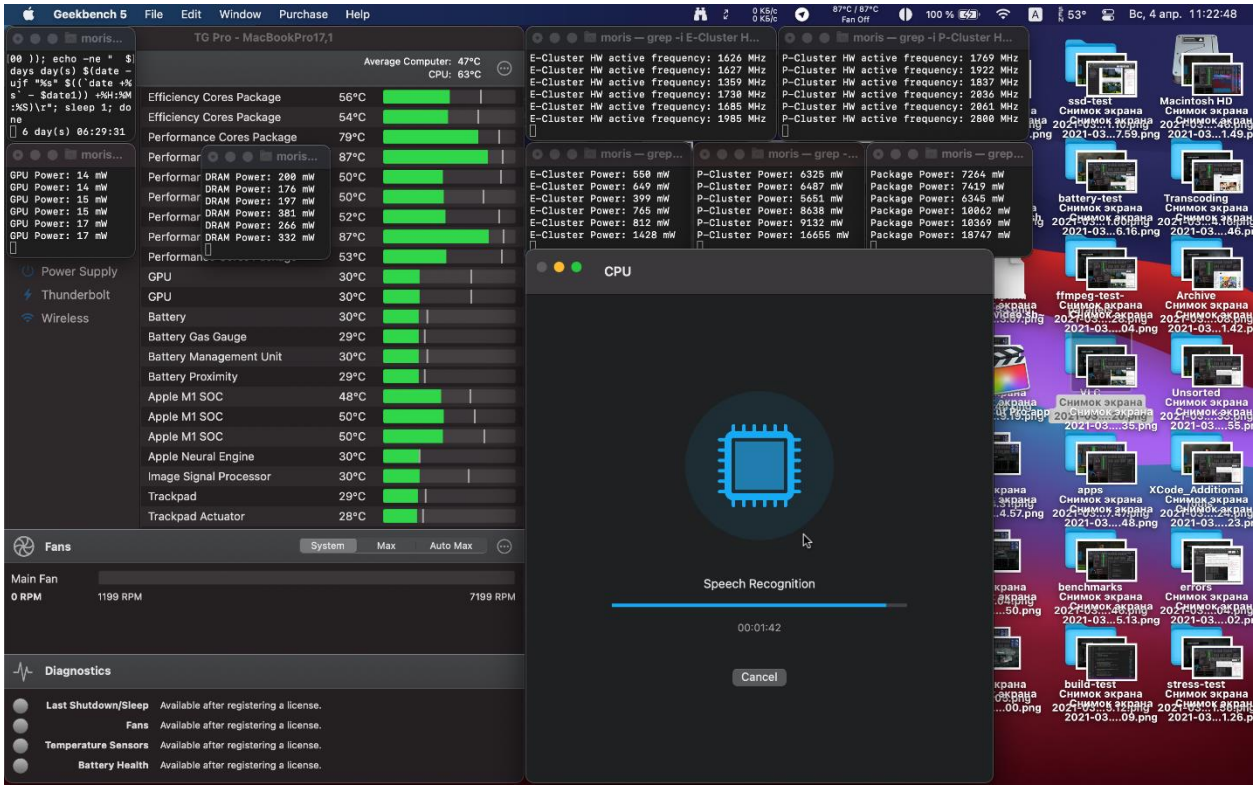


Скриншот 93. Macbook Pro 13 2020 4x TB3 (i5 1038NG7, 33W). Результат теста CPU в Geekbench 5.

Если сравнивать с Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (Core i5 1038NG7, 33W) разница в пользу M1 составила в Multi Core 73% без Rosetta 2 и 37% с ней, а в Single Core – 41% без Rosetta 2 и 8.5% с ней.

Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (Core i5 1038NG7, 33W) на 16% опережает мой Xiaomi в Multi Core и на 23% в Single Core. Но ощутимо ли это? Узнаем через одну подглаву (в 5.5).

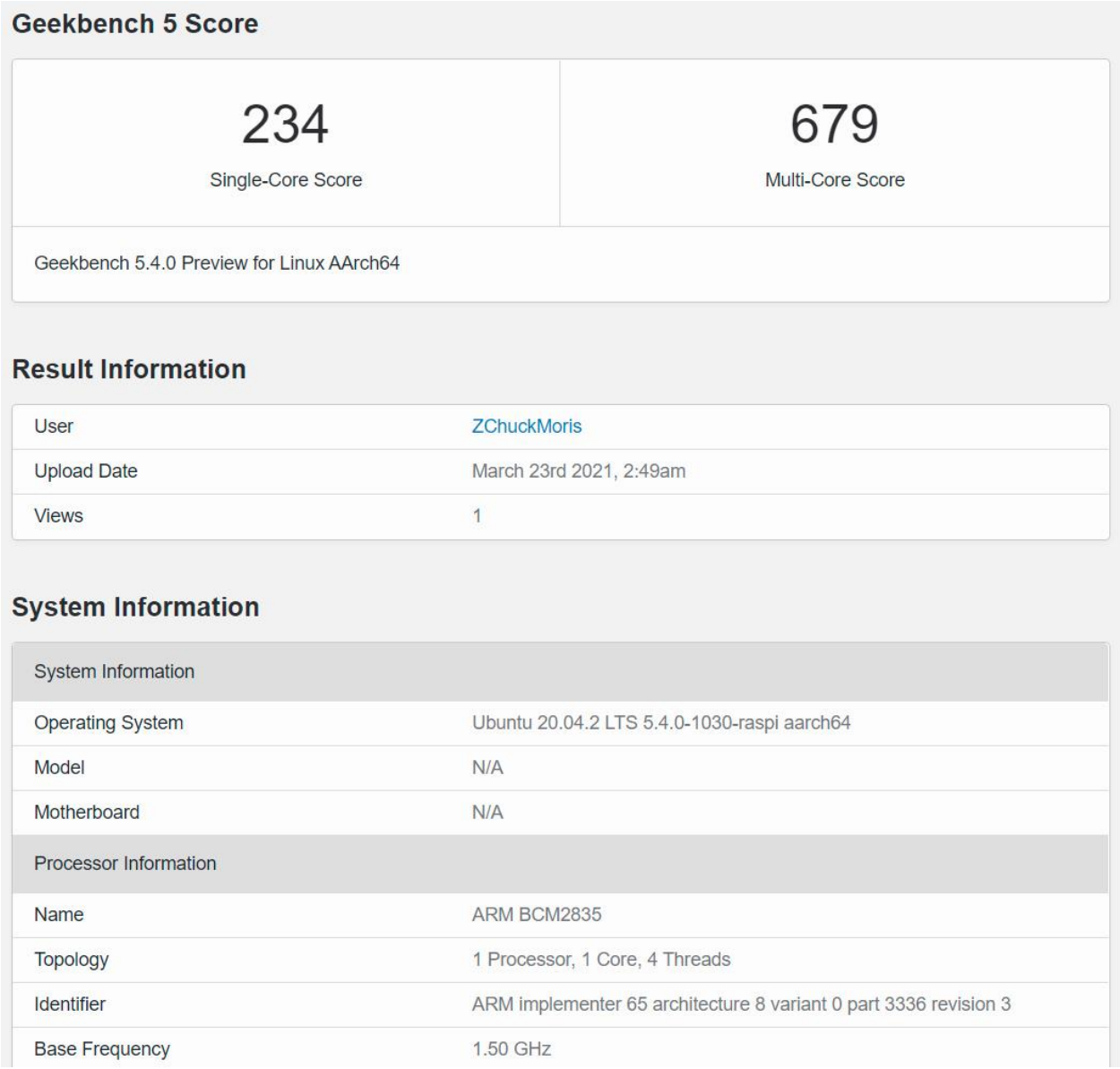
А вот при сравнении Macbook Pro на Apple M1 с Lenovo Yoga Slim 7 с Ryzen 7 4800U в Geekbench 5 выходит, что первый быстрее аж на 9.1%. Что? Как? Вспомните, Ryzen 7 4800U потребляет до 43 Ватт в пике, и тест Geekbench 5 он проходит также менее чем за 3 минуты, т.е. его TDP не успевает снизиться до 25 Ватт. А как много тепла выделяет M1 в Geekbench?



Скриншот 94. Macbook Pro 13 2020 (Apple M1, 20W). Мониторинг тепловыделения CPU в ходе теста Geekbench 5.

19 Ватт в пике. Большую же часть времени Apple M1 в тесте Geekbench 5 потреблял не более 11 Ватт энергии, в среднем его потребление энергии составляло около 7-8 Ватт, и выделял он соответствующее количество тепла. На скриншоте выше видно, что у Macbook Pro на Apple M1 в ходе теста в Geekbench 5 даже кулеры не включились. И при этом он умудрился на 9% опередить Ryzen 7 4800U в Lenovo Yoga Slim 7. С Rosetta 2 же он всего на 14% отстал от Ryzen 7 4800U с тепловыделением в 2+ раза выше. Я в шоке.

Вы сейчас скажете: а, ну классно, пойду и куплю тогда одноплатный микрокомпьютер на ARM и буду жить да не тужить. Фигушки.



Geekbench 5 Score

234	679
Single-Core Score	Multi-Core Score

Geekbench 5.4.0 Preview for Linux AArch64

Result Information

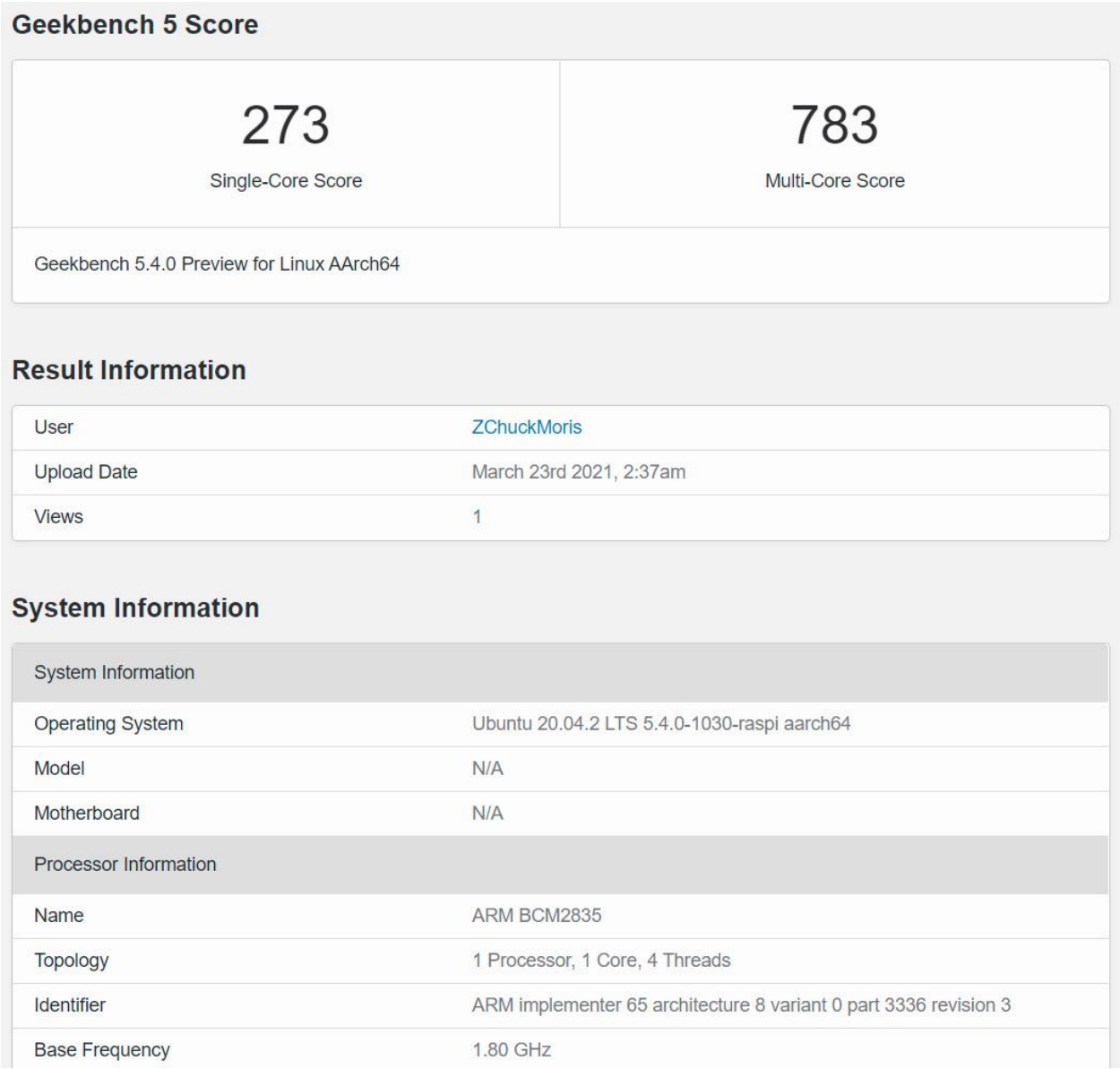
User	ZChuckMoris
Upload Date	March 23rd 2021, 2:49am
Views	1

System Information

System Information	
Operating System	Ubuntu 20.04.2 LTS 5.4.0-1030-raspi aarch64
Model	N/A
Motherboard	N/A
Processor Information	
Name	ARM BCM2835
Topology	1 Processor, 1 Core, 4 Threads
Identifier	ARM implementer 65 architecture 8 variant 0 part 3336 revision 3
Base Frequency	1.50 GHz

Скриншот 95. Результат Raspberry Pi 4 без разгона в тесте Geekbench 5.

Пока я писал эту статью, был представлен Geekbench 5 для систем с ARM64 (aarch64) процессорами под управлением Linux. Я протестировал в ней свою малину (Raspberry Pi 4). Без разгона, т.е. со стандартной тактовой частотой в 1.5 ГГц, результат в одноядерном режиме всего 234 балла, а в многоядерном – 679 баллов. Это в 7.5 и 11.3 раза меньше, чем у Macbook Pro 13 на Apple M1. Это при том, что 4-ая «малина» - одна из самых производительных одноплатных машин на ARM. Да, у неё уже старые ядра Cortex-A72, и у современных смартфонов ядра куда более свежие и производительные, но Вы вряд ли запустите десктопный Linux на смартфоне.



Geekbench 5 Score

273	783
Single-Core Score	Multi-Core Score

Geekbench 5.4.0 Preview for Linux AArch64

Result Information

User	ZChuckMoris
Upload Date	March 23rd 2021, 2:37am
Views	1

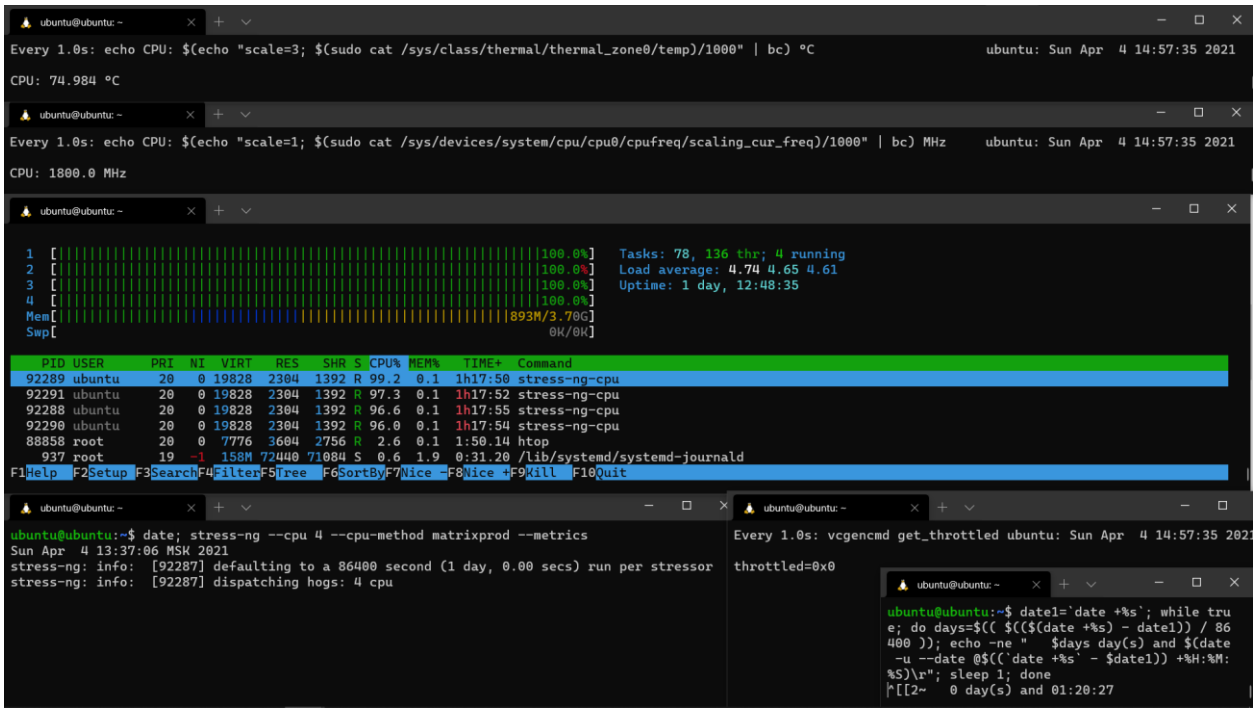
System Information

System Information	
Operating System	Ubuntu 20.04.2 LTS 5.4.0-1030-raspi aarch64
Model	N/A
Motherboard	N/A
Processor Information	
Name	ARM BCM2835
Topology	1 Processor, 1 Core, 4 Threads
Identifier	ARM implementer 65 architecture 8 variant 0 part 3336 revision 3
Base Frequency	1.80 GHz

Скриншот 96. Скриншот 78. Результат Raspberry Pi 4 с разгоном с 1.5 ГГц до 1.8 ГГц в тесте Geekbench 5.

Если разогнать «малину» с 1.5 ГГц до 1.8 ГГц (такая частота у Raspberry Pi 400 по умолчанию), то мы получим производительность центрального процессора (без уч. графики и др. компонентов) чуть меньшую, чем у Qualcomm Snapdragon 820 и 821, которые в своё время устанавливались в Google Pixel (2016), LG G6, Galaxy S7 (версии для США и Китая), Xiaomi Mi 5, HTC 10, OnePlus 3T, Motorola Moto Z, Sony Xperia XZ и др.

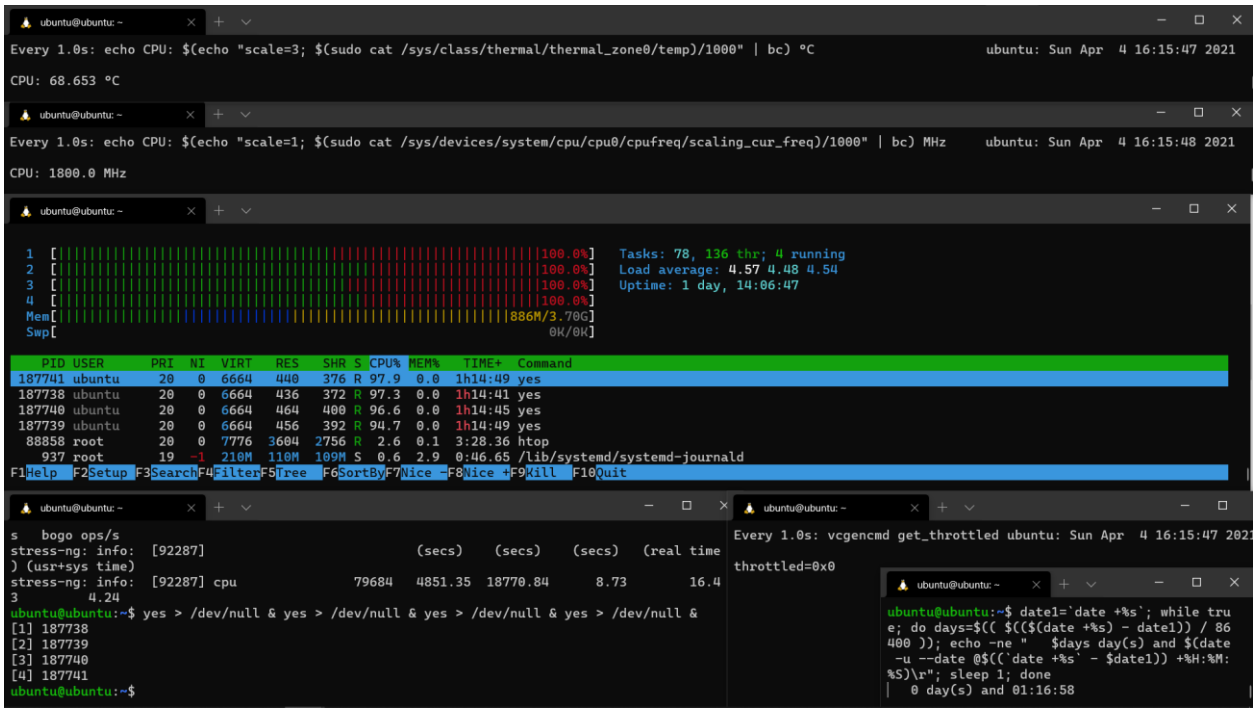
У меня версия Raspberry Pi 4 с 4 ГБ оперативной памяти. С учётом всех моих фоновых служб у неё занято оперативной памяти чуть меньше 1 ГБ, так что дело вовсе не в нехватке оперативной памяти. Просто «малина» в июне 2019 года вышла с SoC Broadcom BCM2711 на базе старых ядер Cortex-A72, в то время как уже вовсю продавались Android-флагманы на базе Cortex-A76.



Скриншот 97. Отсутствие троттлинга на Raspberry Pi 4 с кулером и разгоном с 1.5 ГГц до 1.8 ГГц. Спустя 1.3 часа.

Сразу уточню: тут дело не в перегреве, его нет, просто у решений, которые Вы можете пойти и купить в магазине сходу, производительность сейчас такова. Благодаря кулеру в стресс-тестах даже с разгоном температура «малины» не превышает 76-77 градусов под нагрузкой.

Обратите внимание на скриншот выше: в правом нижнем углу у throttled значение – 0x0. Это значит, что за время тестирования троттлинга не было. Проверял командой **sudo watch -n 1 vcgencmd get_throttled**



Скриншот 98. Стресс-тест с помощью команды Yes на Raspberry Pi 4 не грузит CPU полностью. Спустя 1.25 часа.

Я также проводил стресс-тест командой yes, аналогичному тому, как я проводил стресс-тест на Macbook Pro с Apple M1, однако на Raspberry Pi 4, как и в целом на компьютерах под управлением дистрибутивов Linux или Windows 10 с WSL2 (Windows Subsystem for Linux 2), утилита stress-ng сильнее «прогревает» CPU, нежели это делает команда yes.

Возвращаясь к теме, основная проблема в том, что одноплатных компьютеров на ARM с сопоставимой с Apple M1 производительностью Вы в свободной продаже не найдёте. Производительных процессоров для десктопов, которые можно было бы приобрести отдельно, также нет. Т.е. единственный для Вас вариант пойти и купить ARM процессор, это купить его в составе уже готового ноутбука на Windows.

И в ноутбуках на данный момент используется преимущественно Qualcomm Snapdragon 8cx на базе уже устаревших ядер Kryo 495 Gold, которые являются производными от Cortex A-76 (аналогичные были установлены в Snapdragon 855). Я нашёл тесты Samsung Galaxy Book S на базе этого SoC на портале notebookcheck от того же Andreas Osthoff, которого я уже упоминал выше в троттлинг-тесте, и в Geekbench 5.3 Snapdragon 8cx набрал всего 700 баллов в одноядерном тесте, что в 2.5 раза ниже показателя Macbook Pro на M1, а в многоядерном тесте он набрал 2760 баллов, что аж в 2.78 раза ниже показателя Macbook Pro 13 на Apple M1.

Если вы думаете, что Geekbench не поддерживает просто aarch64 (ARM64) архитектуру на Windows, то вы ошибаетесь, т.к. нативная поддержка Windows 10 on ARM в Geekbench была добавлена ещё с версии 4.3. В добавок, в середине 2020 года, когда была опубликована статья с обзором этого ноутбука на notebookcheck, Windows 10 on ARM ещё не поддерживала эмуляцию x86_64 процессора, поддерживалась лишь эмуляция 32-битной x86 системы, т.к. возможность эмуляции x86_64 была добавлена лишь недавно обновлением Insider Preview версии Операционной Системы Windows 10 on ARM от Microsoft, а в статье тесты Geekbench 5 проводились в 64-битном режиме.

Какие ещё есть процессоры для ноутбуков на ARM, которые уже имеются в продаже? Остаётся только Microsoft SQ1, который по факту является тем же Snapdragon 8cx, но с частотой на 160 МГц (5.6%) выше.

Я нашёл его результаты в Geekbench 5, они не сильно то и отличаются.

Что ещё остаётся? Snapdragon 8cx Gen 2, у которого ещё чуть выше частота, уже на 310 МГц (11%) выше в сравнении с 8cx 1-го поколения, но в целом то микроархитектура та же, ядра те же, производительность будет +- та же.

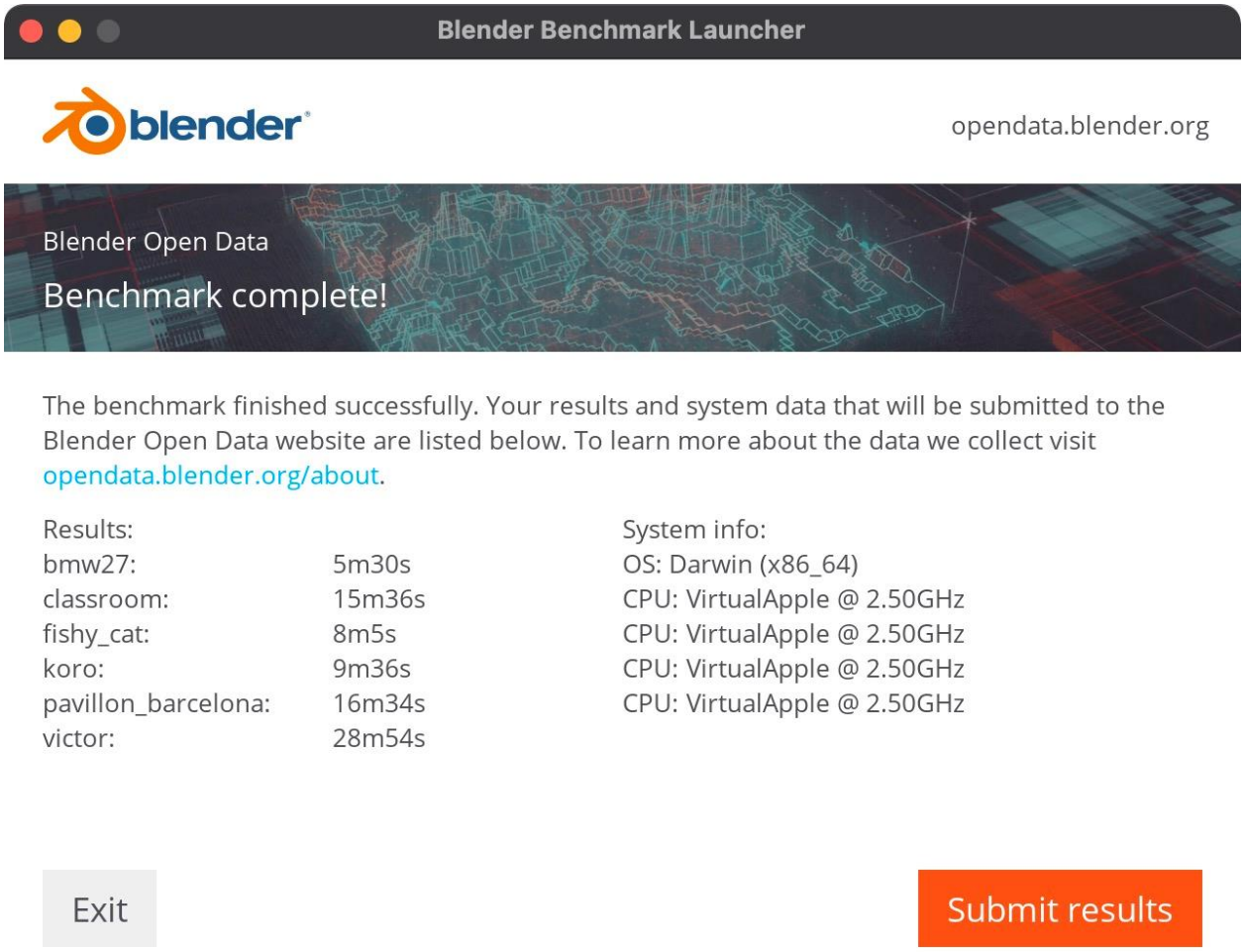
Куда деваться, сравним тогда со свежими процессорами от Qualcomm в смартфонах, а именно – Snapdragon 888. Открываем видео «SNAPDRAGON 888 vs EXYNOS 2100— Вы удивитесь!» на канале Droider, таймкод – 2:38. У Galaxy S21 со Snapdragon 888 в Geekbench 5 результат – 3651 баллов в Multi Core и 1132 балла в Single Core, а у модели с Exynos 2100 – 3305 и 1055 баллов соответственно. Первый отстаёт Macbook Pro с Apple M1 в 2,1 раза в Multi Core и в 1,54 раза в Single Core, а 2-ой – в 2,32 раза в Multi и 1,65 раза в Single. И то в Single Core отставание новых SoC Qualcomm и Exynos не так велико только потому, что у них имеется одно «жирное» альфа-ядро Cortex-X1, которое по производительности заметно мощнее остальных ядер Cortex-A78 в составе CPU. Но это ядро не особо поможет в задачах, когда загружено не 1 ядро, а 2 или 4. Конкурентами Apple M1 эти SoC никак не назвать.

Но ведь среди производителей CPU на ARM есть ещё Huawei, верно? Открываем видео Стаса «БЕЗ GOOGLE ЗА 100 000 РУБЛЕЙ? – Месяц с Huawei Mate 40 Pro [Честный обзор]», таймкод – 8:22. Результат Mate 40 Pro с Kirin 9000 в том же Geekbench 5: 3532 балла в Multi Core и 986 баллов в Single Core. Аналогично: слабее в 2,17 раза и в 1,77 раза соответственно...

Так и получается, что на потребительском рынке другие равные аппараты на ARM нам неоткуда достать. Таких нет. И даже если попадётся Вам ноутбук со Snapdragon 888, он не будет ровней даже Apple A14 с 2 ядрами вместо 4, т.к. Windows, в отличие от macOS, использует эмуляцию платформы, а не трансляцию кода с одной архитектуры в другую, что я разбирал в главе 2.1. В x86 приложениях производительность будет так себе.

5.4. Blender 2.91.2 и Blender 2.92.

В главе 2.2 я упоминал, что Blender 2.90 на M1 не удаётся запустить на Mac с Apple Silicon из-за ошибки в проверке на AVX инструкции в OIDN, одной из составных библиотек Blender. Поэтому тестируем 2.91.2 и 2.92.

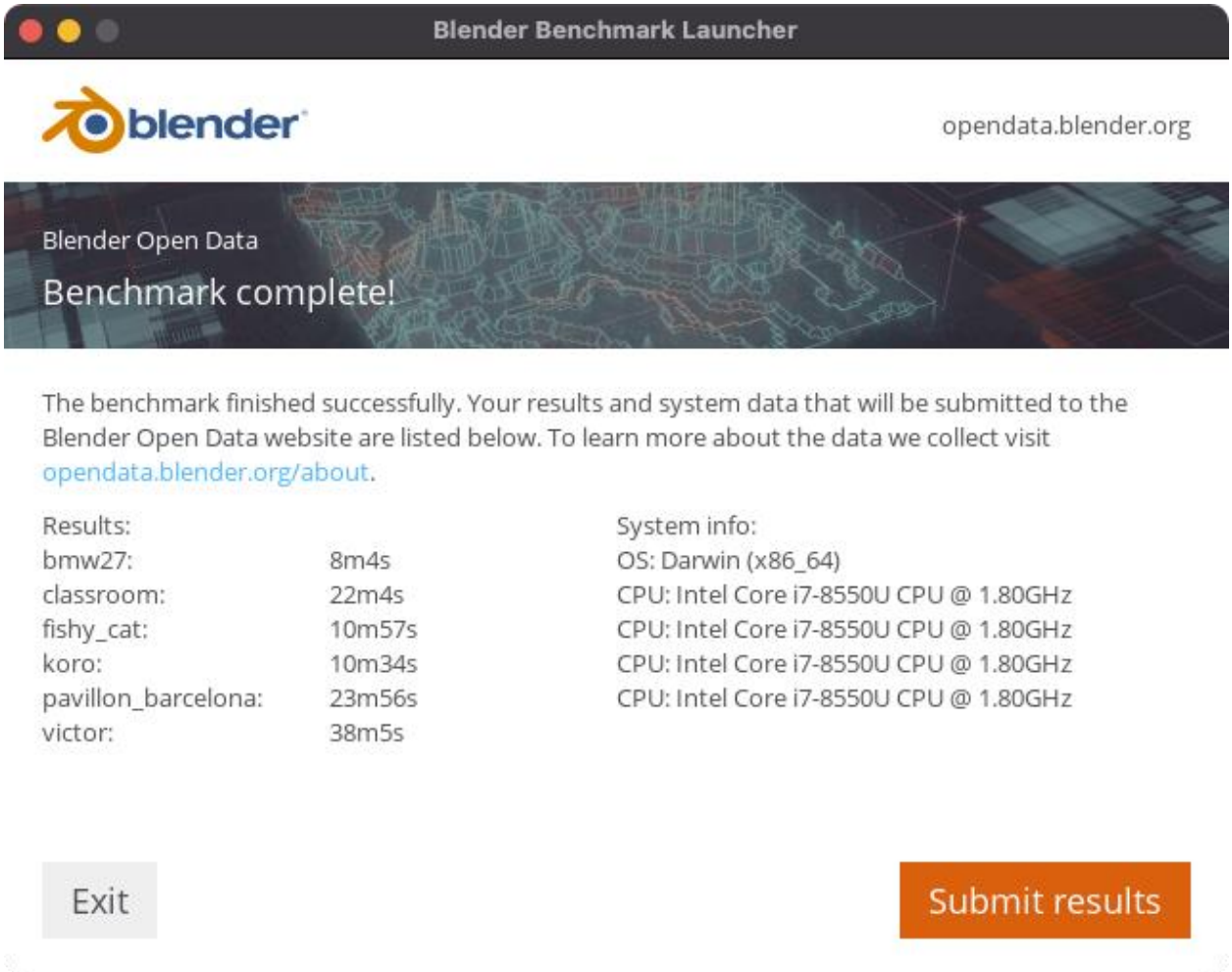


Скриншот 99. Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Blender 2.91.2. Rosetta 2 (Intel mode).

Вы только вдумайтесь в этот абсурд. Ryzen 7 4800U в режиме максимальной производительности при тепловыделении в первые 3 минуты около 30-32 Ватт в среднем и дальнейшем тепловыделении в 25 Ватт, при использовании AVX-инструкций, при том, что Blender ещё и не собран под Apple M1 нативно, т.е. работает только через предварительную трансляцию через Rosetta 2, опередил Apple M1 в Macbook Pro 13 всего на 21,3% в Blender 2.91.2 по средней скорости рендеринга всех сцен (просто сложите время рендеринга всех сцен на M1 и поделите на время рендеринга Ryzen 7 4800U, и отнимите от этого единицу). Тут все условия были в пользу AMD, но разница по итогу оказалась сопоставима с разницей в энергопотреблении.

Разница в сцене koro и вовсе составила в пользу AMD всего 2.5%. Если же сравнивать Macbook Pro с Apple M1 и Lenovo Yoga Slim 7 с Ryzen 7 4800U уже в режиме интеллектуального охлаждения, когда у Lenovo TDP опускается до 15-20 Ватт (зависит от задачи), разница между ними составляет 2.4% в пользу Lenovo по средней скорости рендеринга всех сцен.

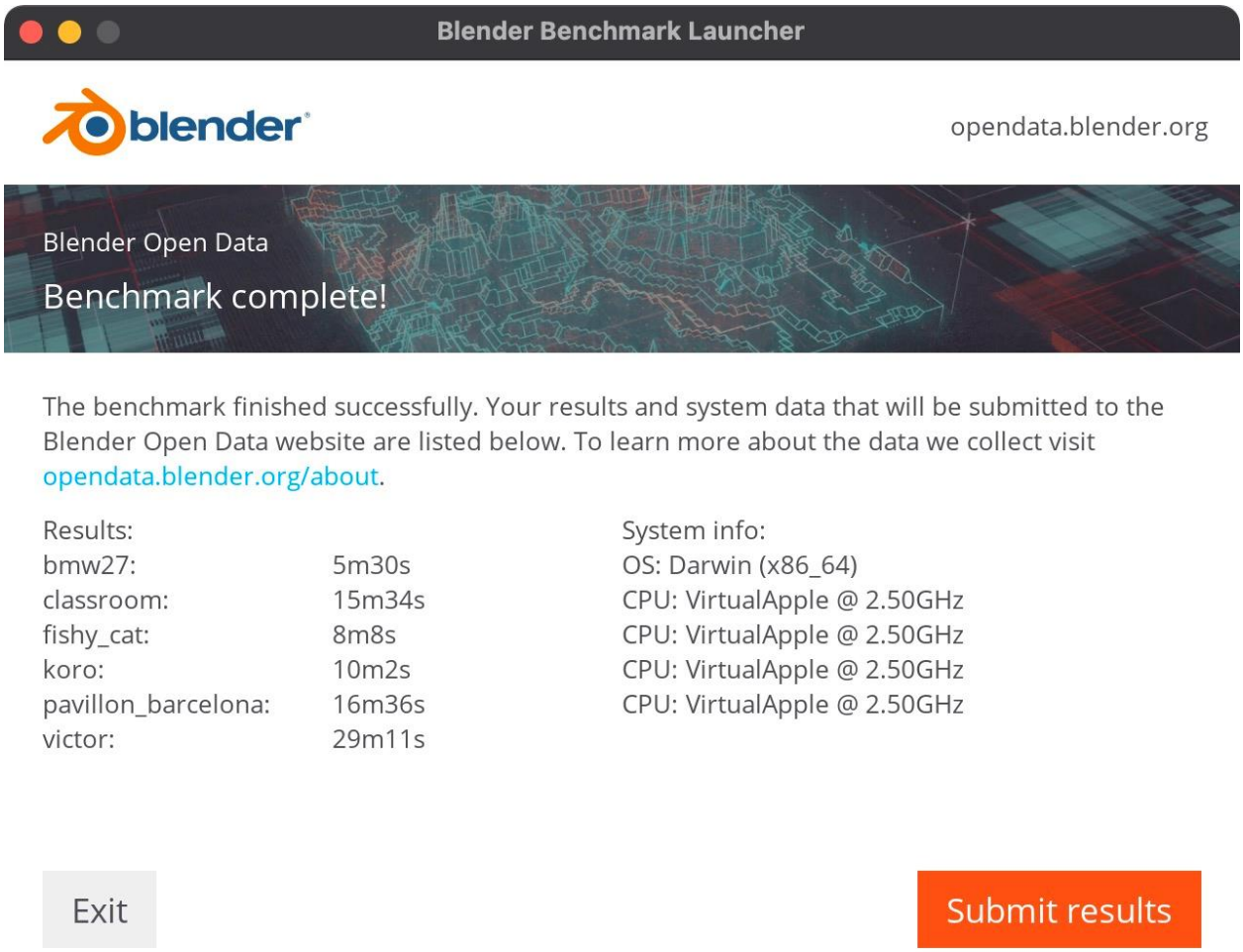
Компания AMD могла бы вырваться вперёд ещё немного, если бы могла производить чипы на 5 нм, но на данный момент для AMD это нереализуемо, т.к. одна лишь Apple забронировала у TSMC около 80% мощностей по производству 5 нм чипов. Ryzen на 5 нм выйдут такими темпами лишь в 2022 году, т.к. у TSMC не осталось свободных мощностей.



Скриншот 100. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Результат теста CPU в Blender 2.91.2.

Разница же между M1 и i7 8550U в Xiaomi составила от 32% до 49% (кроме сцены koro, там разница составила всего 8,56%) в пользу M1.

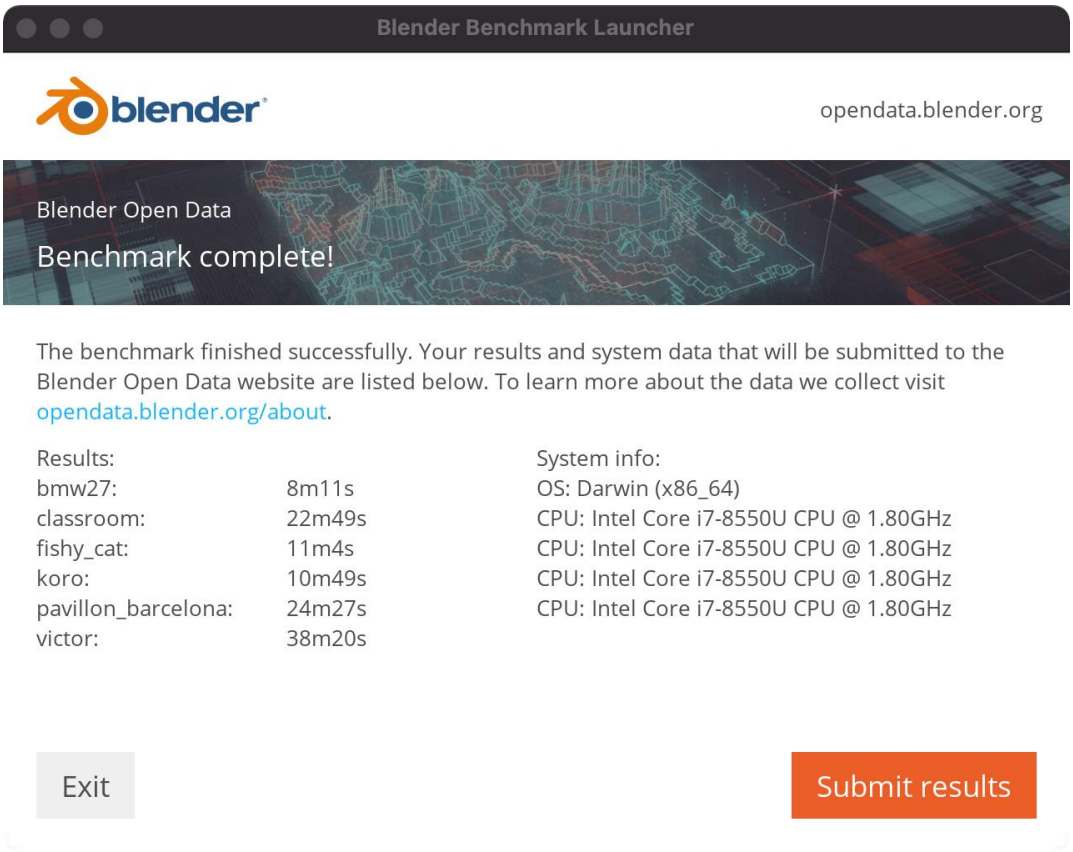
Пока я писал эту часть статьи, вышла новая версия Blender, 2.92.



Скриншот 101. Apple Macbook Pro 13 2020 (Apple M1, 20W). Результат теста CPU в Blender 2.92.

С версией 2.92 результаты у Macbook Pro с SoC Apple M1 в среднем чуть хуже, чем с 2.91.2 (в среднем разница 0.5%), выделяются koro (3.6%) и bmw27 (2.7%), но это укладывается в пределы погрешности, потому учитывать эту разницу мы особо не будем.

При сравнении Macbook Pro с Apple M1 с Lenovo Yoga Slim 7 с Ryzen 7 4800U всё также: отрыв Lenovo составил всего 23.3% в режиме максимальной производительности по средней скорости рендеринга всех сцен, а в сцене koro отрыв Lenovo составил всего 7.3%. Это больший отрыв, чем тот, что был с версией Blender 2.91.2, но это и близко не уровень Macbook Pro 13 2020, против которого тут играет всё: и исполнение в режиме Rosetta 2 на нём без NEON/SVE, и AVX у конкурента, и буст у конкурента до 43 Ватт в первые секунды, и среднее тепловыделение в 30-32 Ватта в первые 3 минуты у конкурента. В режиме интеллектуального охлаждения же отрыв Lenovo Yoga Slim 7 составил всего 2.8% в среднем. Вообще ни о чём.



Скриншот 102. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Результат теста CPU в Blender 2.92.

У Xiaomi с Blender версии 2.92 аналогично маку на M1 результат ниже на 1,85% в среднем, выделяются только сцена Classroom (3,4%). В целом разница же укладывается в пределы погрешности у обоих ноутбуков.

В общем, выходит, что производительность в Blender у нового Macbook довольна высока, даже при том, что Blender ещё не был выпущен под macOS для ARM, т.е. даже с Rosetta 2 новый Macbook с чипом M1 оказывается столь феноменально производительным для своих рамок тепловыделения. Вопрос: а задействовала ли Rosetta 2 расширения инструкций SVE при трансляции SSE кода Blender с x86 на ARM? Если да, то насколько эффективно? Если же нет, мне тяжело поверить в то, что даже без SVE чип Apple M1 не сильно уступает Ryzen 7 4800U, который задействовал для той же операции AVX инструкции. Ещё бы охлаждение нормальное поставили, или просто оставили бы его нетронутым с прошлого поколения Macbook Pro 13 2020 с 2x TB3 с чипом Intel Core i5 8257U, и вопросов бы не было у меня.

А теперь перейдём к тестам, результаты которых вряд ли Вам удастся найти в сети. Это лишь некоторые тесты из тех, которые хотели видеть в этой статье я и немногие из тех 7 человек, которых я упоминал в начале статьи.

5.5. Скорость компиляции Swift проектов для Mac в XCode.

В сети есть тесты с тестами скорости компиляции проектов в XCode, но у них есть одна весьма знатная проблема: во всех тестах, что я находил, компилировали только проекты под iOS в обоих случаях, и всегда побеждал Macbook с чипом M1. А знаете почему? Да потому, что компиляция велась под его родную платформу (ARM -> ARM), а в случае с макбуками на Intel велась кросс-компиляция (Intel -> ARM), которая сильнее грузит систему. Очевидно, победит Macbook на M1, к гадалке не ходи, отсюда и выводы, что M1 быстрее, чем Core i9. Ну, это надо проверять то не в одной задаче, верно?

Вторая проблема этих тестов в том, что либо проверялась скорость сборки проекта с закрытым исходным кодом (т.е. вы не сможете загрузить себе исходный код и протестировать всё то же самое самим), либо тест проводился с проектом XCodeBenchmark. Его, помимо канала Max Tech, для которого он и проектировался, использовали также Linus Tech Tips на 11:54 и Alexander Ziskind. XCode Benchmark – это просто «сборная солянка» из множества разных библиотек для сборки проекта под iOS. Этот проект не имеет ничего общего с реальной разработкой. Да и разве можно заявлять о превосходстве M1 над Intel только на основе сравнения нативной сборки (ARM -> ARM) на M1 в сравнении с кросс-компиляцией (x86_64 -> ARM)? В главе 5.6 тесты компиляции под iOS мы также проведём, но другие.

Для сборки мы используем XCode версии 12.4 (свежайший на сегодня).

Какой же реальный проект мы будем собирать под Mac (x86+ARM)?

Медиаплеер IINA. Довольно неплохой плеер с открытым исходным кодом. С его кодом можете ознакомиться на странице репозитория в GitHub.

Как мы его будем собирать? Для начала Вам потребуется установить cocoapods, чтобы загрузить все зависимости. Есть 2 способа его поставить:

1. Через gem. Чуть более геморный метод. Через gem cocoapods ставятся пока только с использованием Rosetta 2 (либо запускаем Terminal с Rosetta 2, либо используем префикс arch в командах):

```
arch -x86_64 gem install cocoapods
arch -x86_64 gem install ffi
arch -x86_64 pod install (это используем уже в папке iina).
```

1. Через brew. Я предпочёл этот метод, т.к. он нативный для ARM.

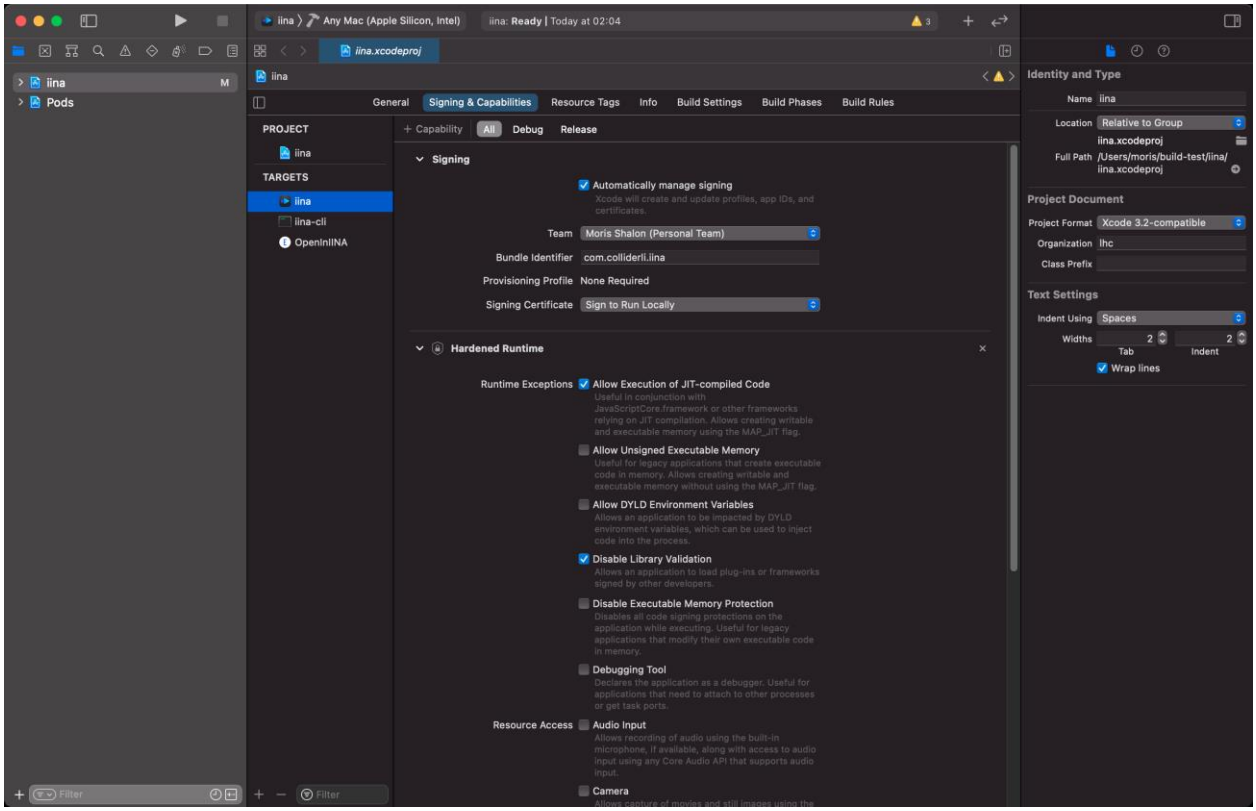
```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install cocoapods libffi
```

Далее, если Вы успешно установили cocoapods, выполняем следующую команду для загрузки исходников и подготовки проекта.

```
for DIR in "$HOME/build-test/iina" "$HOME/Library/Developer/XCode/DerivedData";
do if [ "$(ls -A $DIR | grep -v '.DS_Store')" ]; then rm -rf $DIR/*; fi; done; cd
~/; mkdir build-test; cd build-test; git clone --recursive
https://github.com/iina/iina.git ; cd iina; git reset --hard
c5b08c443d6c2c91604f3ab0099fe51440e84644; pod install; ./other/download_libs.sh;
open iina.xcworkspace
```

Этот набор команд загрузит Вам исходный код программы в папку ~/build-test/ (создастся папка build-test в директории пользователя и туда всё загрузится), и приведёт проект к версии 1.2.0, собираем мы именно её. Далее будут загружены все зависимости, используемые проектом и запущен XCode.

В XCode всё вполне просто. Ждём, пока проект полностью индексируется. В Signing and capabilities указываем в Team свой аккаунт. Можно для локального личного использования задействовать и обычный iCloud, и тогда у Вас будет значиться "Personal Team". Вам не обязательно платить ради теста этой сборки 100$ в год за сертификат разработчика.

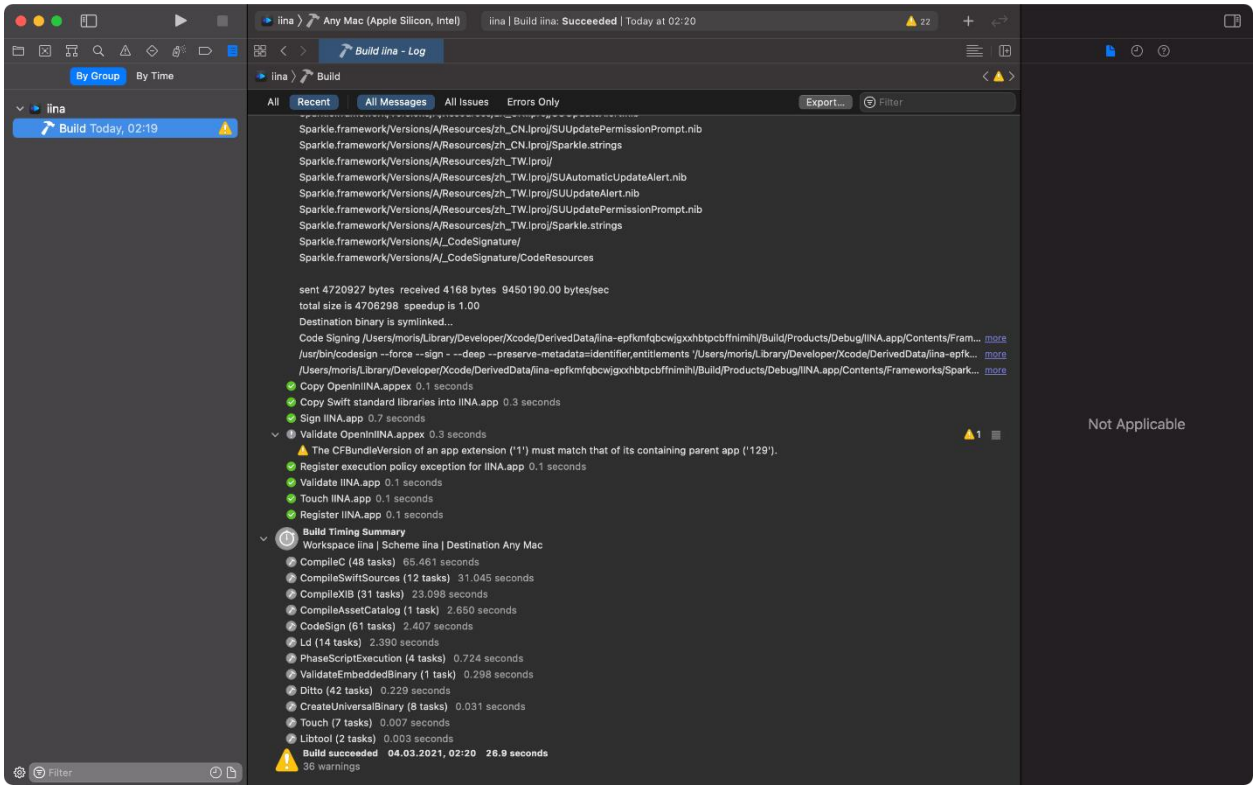


Скриншот 103. Настройка проекта IINA для сборки в XCode под x86+ARM.

В XCode в верхней части центрального окна выбираем iina в Targets, и Any Mac (Apple Silicon, Intel) в Build, как в верхнем левом углу на скриншоте выше. В проекте в Signing and Capabilities: удаляем App Groups, iCloud и in-App Purchase. Также в Bundle Identifier добавляем в конце -test-любой-номер. Суть в том, что у вас должен быть уникальный Bundle Identifier для сборки проекта без учётной записи разработчика за 100$/год.

После этого жмём в Menu Bar на Product - Perform Action - Build with timing summary.

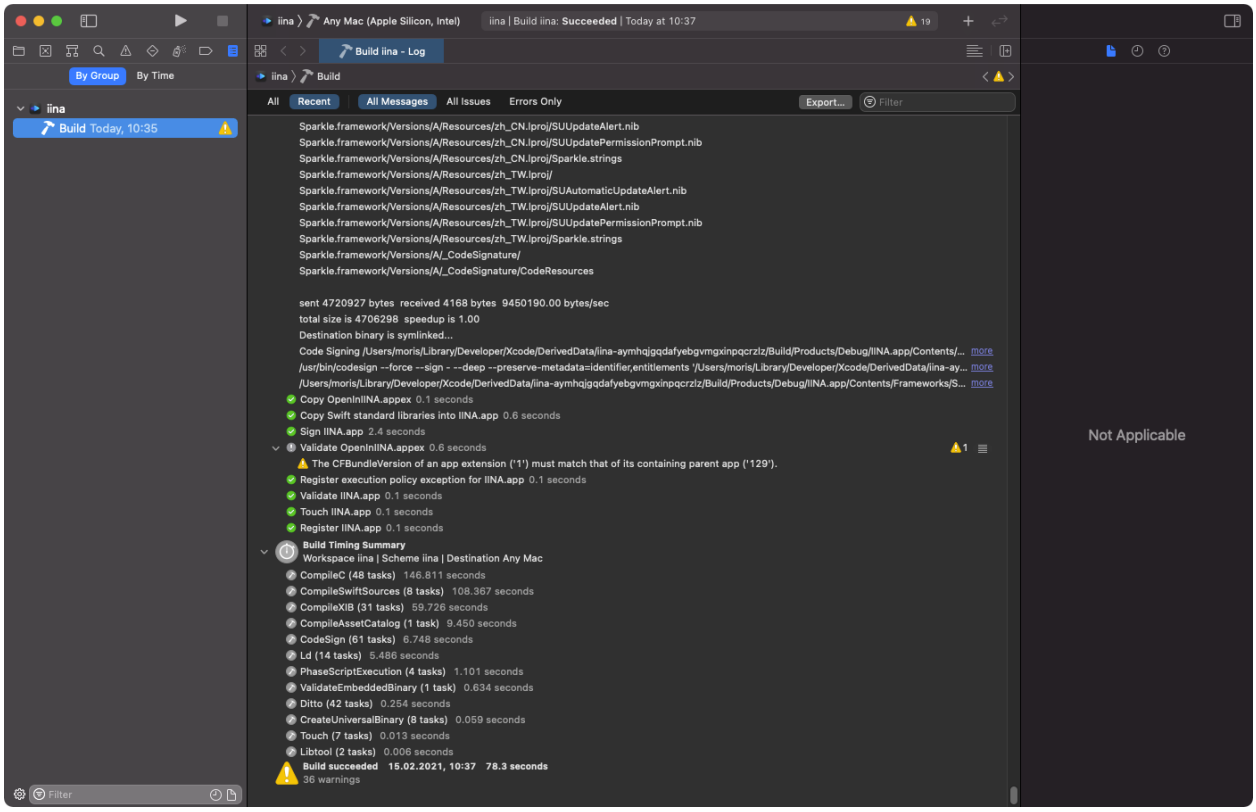
Когда проект полностью собрался, в панели слева нажимаем на "Show the report navigator" (крайняя иконка справа), выбираем By Group, выбираем свой процесс Build в iina и в центральной вкладке выбираем Recent вместо All. В конце вывода XCode находим время, за которое проект был собран. Тут и полное время сборки, и время, которое было бы затрачено на сборку каждого отдельного компонента, если бы он собирался в 1 поток.



Скриншот 104. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта IINA в XCode.

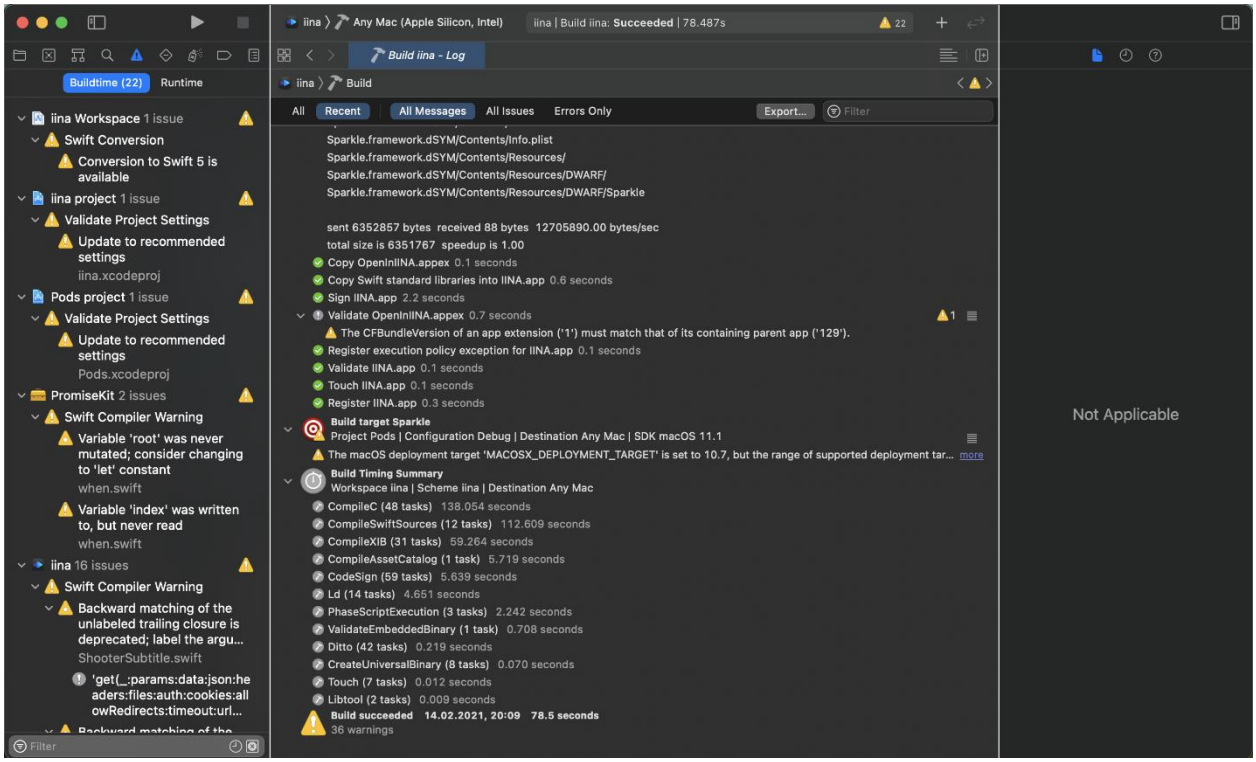
И вот мы видим, что проект собрался за 26.9 секунд под обе платформы (x86+ARM). А много это или мало? По идее, тут не должно быть тотального доминирования M1, т.к. что в случае с Macbook на Apple Silicon,

что в случае с Macbook на Intel, тут выполняются как нативная компиляция под свою платформу, так и кросс-компиляция.

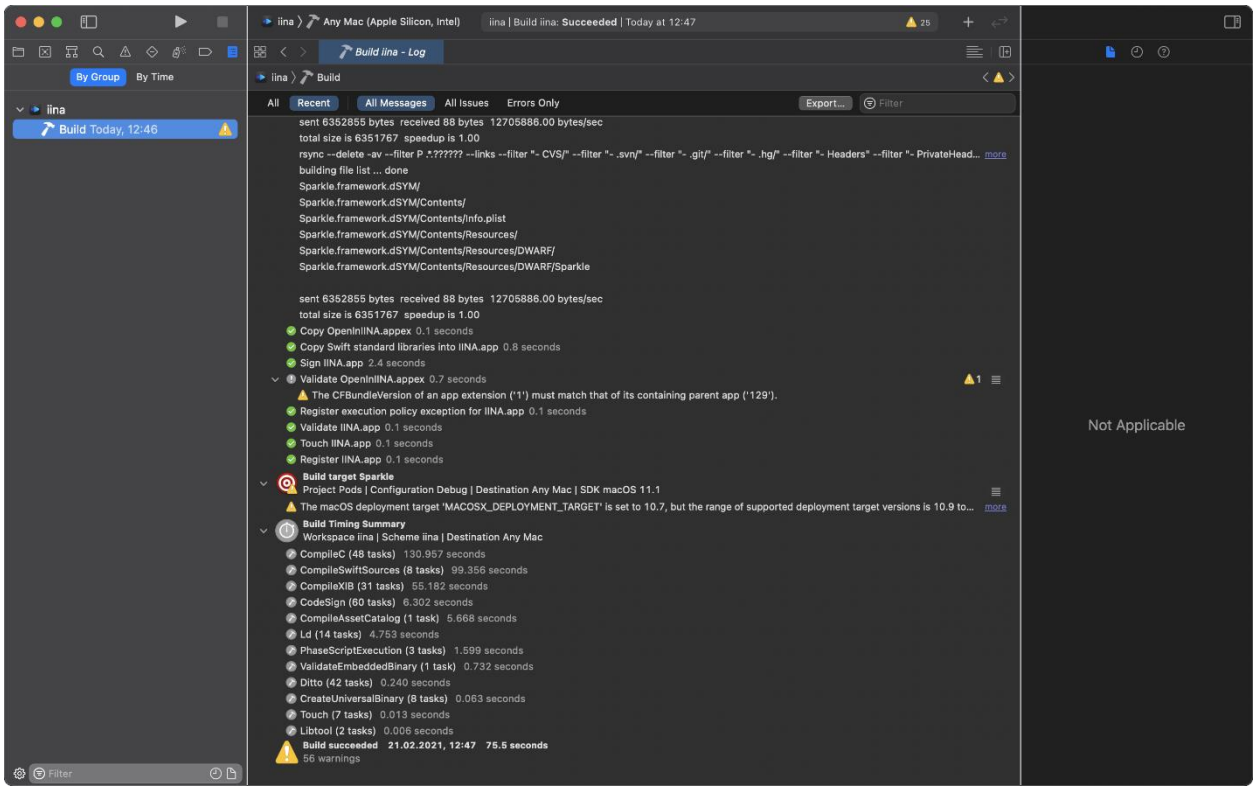


Скриншот 105. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция проекта IINA в XCode.

Хочется сравнить с моим Xiaomi, но странно. Разница составляет целых 2.91 раза, но ведь не может быть, верно? Сравним с Macbook на Intel.



Скриншот 106. Macbook Pro 13 2018 4x TB3 (i5 8259U, 33W). Компиляция проекта IINA в XCode.



Скриншот 107. Macbook Pro 13 2020 4x TB3 (i5 1038NG7, 33W). Компиляция проекта IINA в XCode.

Выше Вы видите также и результаты 2-ух макбуков с 4 портами Thunderbolt 3 и чипом i5 (модели 2018 и 2020 годов). Это уничтожение. У Macbook Pro 13 2018 результат практически 1 в 1 как у моего Xiaomi, там отличие всего в 0.2 секунды, что запросто списываем на погрешность, но и у Macbook Pro 13 2020 результат то недалеко ушёл от моего Xiaomi, разница всего 3.7%. Как видите, та разница в 15%, которую мы наблюдали в тесте Cinebench ранее, тут нигде не ощущается. Зато преимущество Macbook Pro на M1 по части производительности тут раскрылось в полной мере. Разница в 191% это - то, чего я никак не мог ожидать. Мы ведь собираем проект и под x86, и под ARM, т.е. кросс-компиляция и нативная компиляция тут проходят в обоих случаях и у M1 из-за архитектуры тут попросту не может быть преимущества.

Напоминаю, что суть нашего теста в том, что вы легко можете его повторить и проверить данные на своём оборудовании. Просто повторяйте, описанные выше, действия и Вы сами удостоверитесь в результатах. Если вдруг я где-то ошибся в описании процесса или что-то упустил, напишите в комментариях, я дополню методику, и в следующих тестах я постараюсь избежать озвученной ошибки.

5.6. Скорость компиляции Obj-C + Swift проектов для iOS в XCode.

Под iOS мы будем собирать из исходного кода проект медиаплеера VLC (VLC-iOS). Если верить GitHub, этот проект на 53% состоит из Obj-C кода и на 34% из Swift кода. То, что надо для теста Obj-C + Swift.

В целом тут процедура схожа с тем, что Вы видели выше при сборке IINA, но я распишу всё в деталях.

Сперва устанавливаем cocoapods. Тут всё схоже с тем, что было в случае с подготовкой к сборке IINA, так что просто скопирую оттуда вариант установки через Brew:

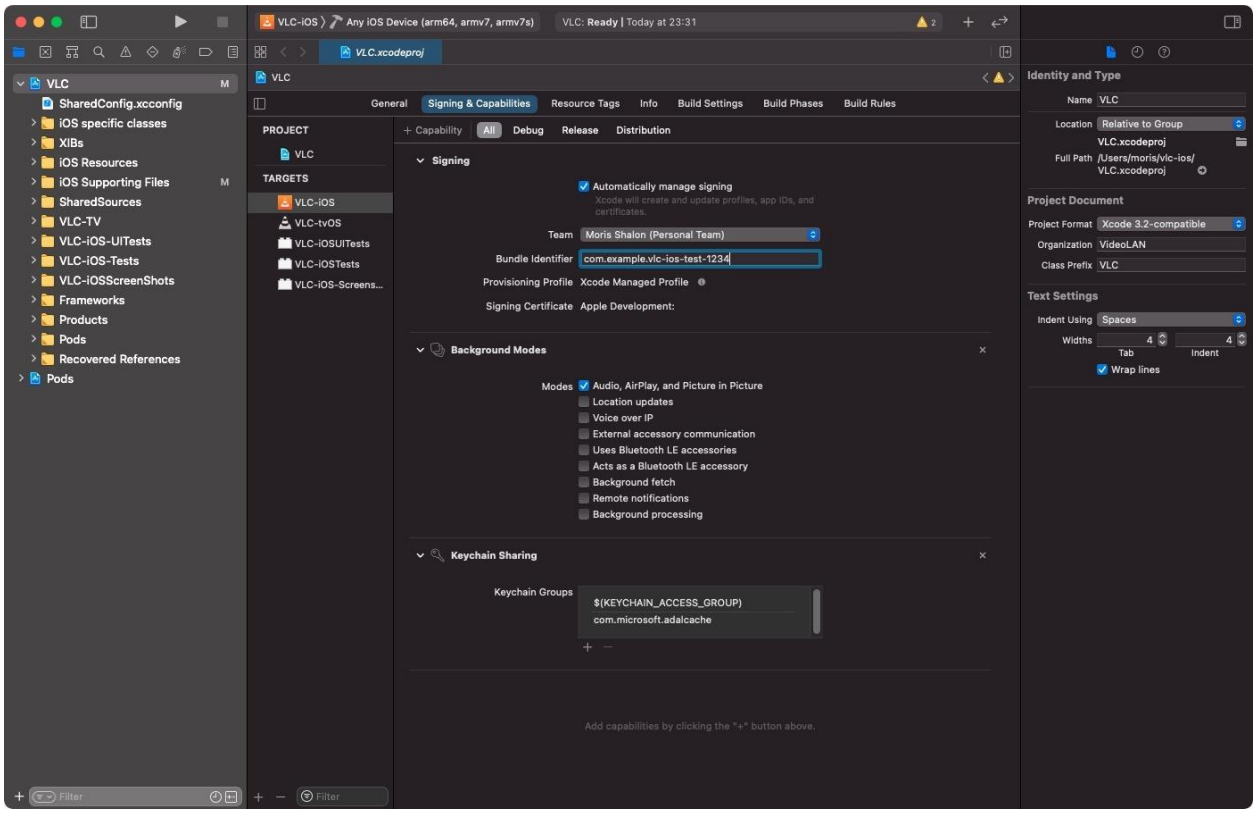
```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install cocoapods
```

libffi в случае с VLC-iOS не требуется, в отличие от IINA, потому его тут не устанавливаем через brew.

Далее для того, чтобы загрузить весь исходный код VLC-iOS, установить все зависимости, очистить кэши XCode, «подтянуть» ту же версию исходного кода, которую я использовал для сборки, и, наконец, запустить проект в XCode, выполните следующий набор команд:

```
CoreSimulatorCachesdyld=$(ls -A
/Users/moris/Library/Developer/CoreSimulator/Caches/dyld | grep -v '.DS_Store');
for DIR in "$HOME/build-test/vlc-ios" "$HOME/Library/Developer/XCode/DerivedData"
"$HOME/Library/Developer/CoreSimulator/Caches/dyld/$CoreSimulatorCachesdyld"; do
if [ "$(ls -A $DIR | grep -v '.DS_Store')" ]; then rm -rf $DIR/*; fi; done; cd
~/; mkdir build-test; cd build-test; git clone --recursive
https://code.videolan.org/videolan/vlc-ios.git; cd vlc-ios; git reset --hard
dc6ba74b0b3fa2ed315ec6dfab3e4415a1f59233; pod install; open VLC.xcworkspace
```

Ждём, пока проект полностью индексируется. В Signing and capabilities указываем в Team свой аккаунт. Можно для локального личного использования задействовать и обычный iCloud, и тогда у Вас будет значиться "Personal Team". Вам не обязательно платить ради теста этой сборки 100$ в год за сертификат разработчика. Единственное дополнительное ограничение в сравнении с IINA в таком случае - Вам нужно будет подключить iPhone/iPad/iPod с iOS 9 или выше, чтобы он подтянулся в XCode и Вам была разрешена сборка под него.



Скриншот 108. Настройка проекта VLC-iOS для сборки в XCode.

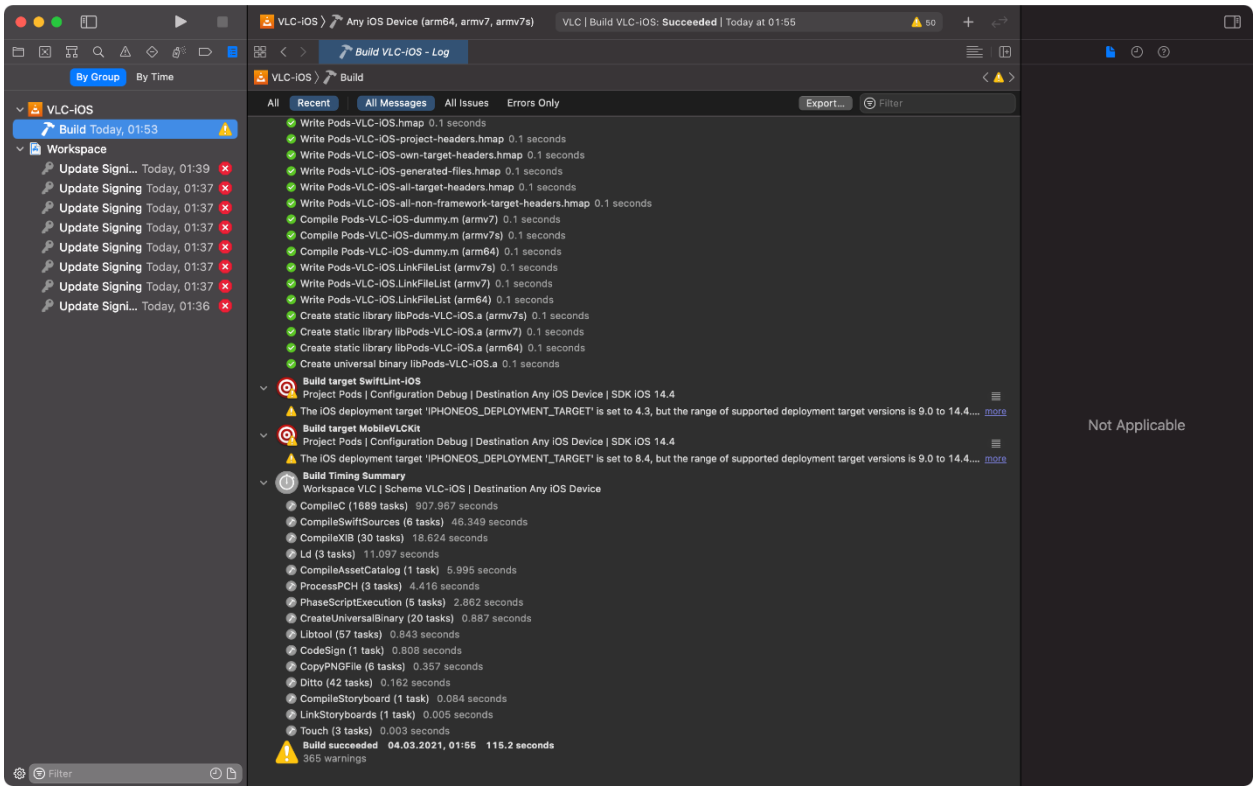
В XCode в центральной вкладке выбираем в Targets VLC-iOS, в нём в Signing and Capabilities: удалить App Groups, iCloud и in-App Purchase. Также в Bundle Identifier добавляем в конце -test-любое_уникальное_значение

Вверху в самой верхней строке центрального окна в VLC-iOS выбираем target в Build – «Any iOS Device (arm64, armv7, armv7s)».

Ориентируйтесь на скриншот выше. По понятным причинам, в «Signing Certificate: Apple Development:» свой почтовый адрес потёр.

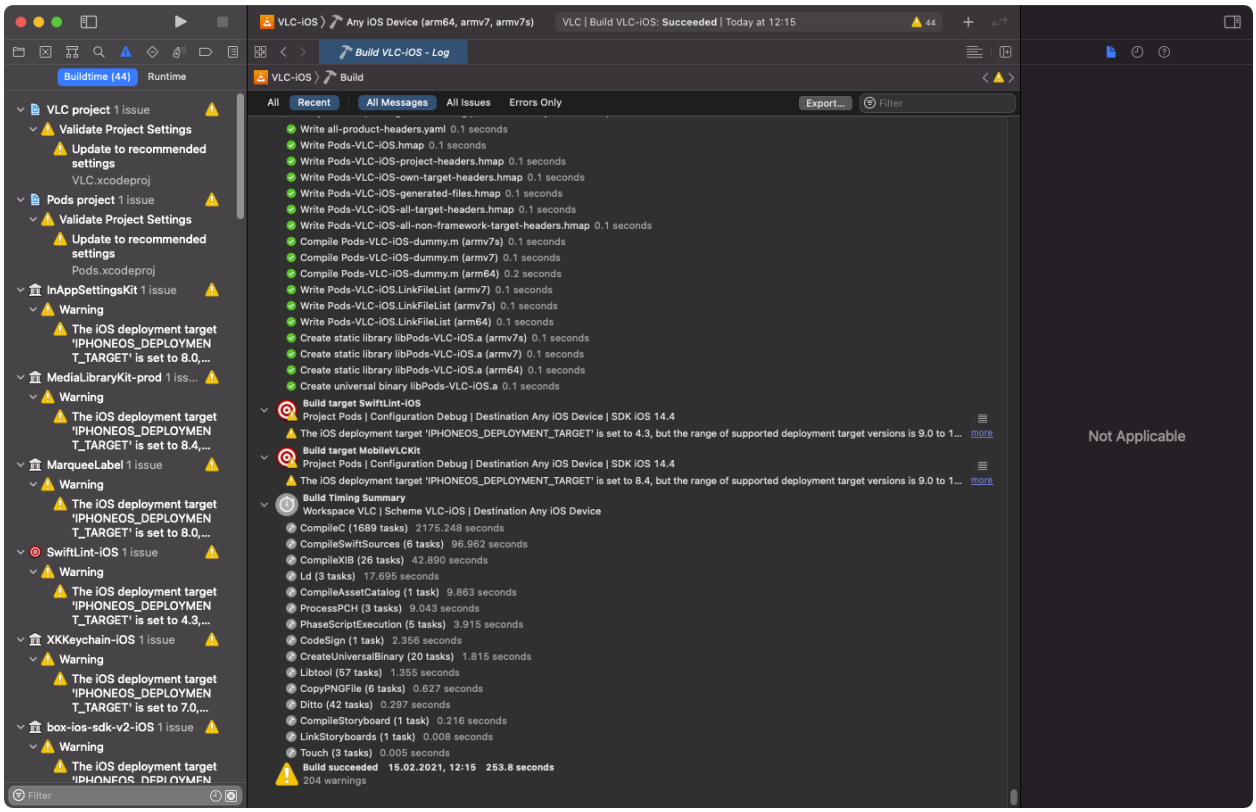
После этого нажимаем в Menu Bar на Product - Perform Action - Build with timing summary. Когда проект полностью собрался, нажимаем на "Show the report navigator" (крайняя иконка справа), выбираем By Group, выбираем свой процесс Build в VLC-iOS и в центральной вкладке выбираем Recent вместо All. В конце вывода находим время, за которое проект был собран. Тут и полное время сборки, и время, которое было бы затрачено на сборку каждого отдельного компонента, если бы сборка велась в 1 поток.

Что в итоге?



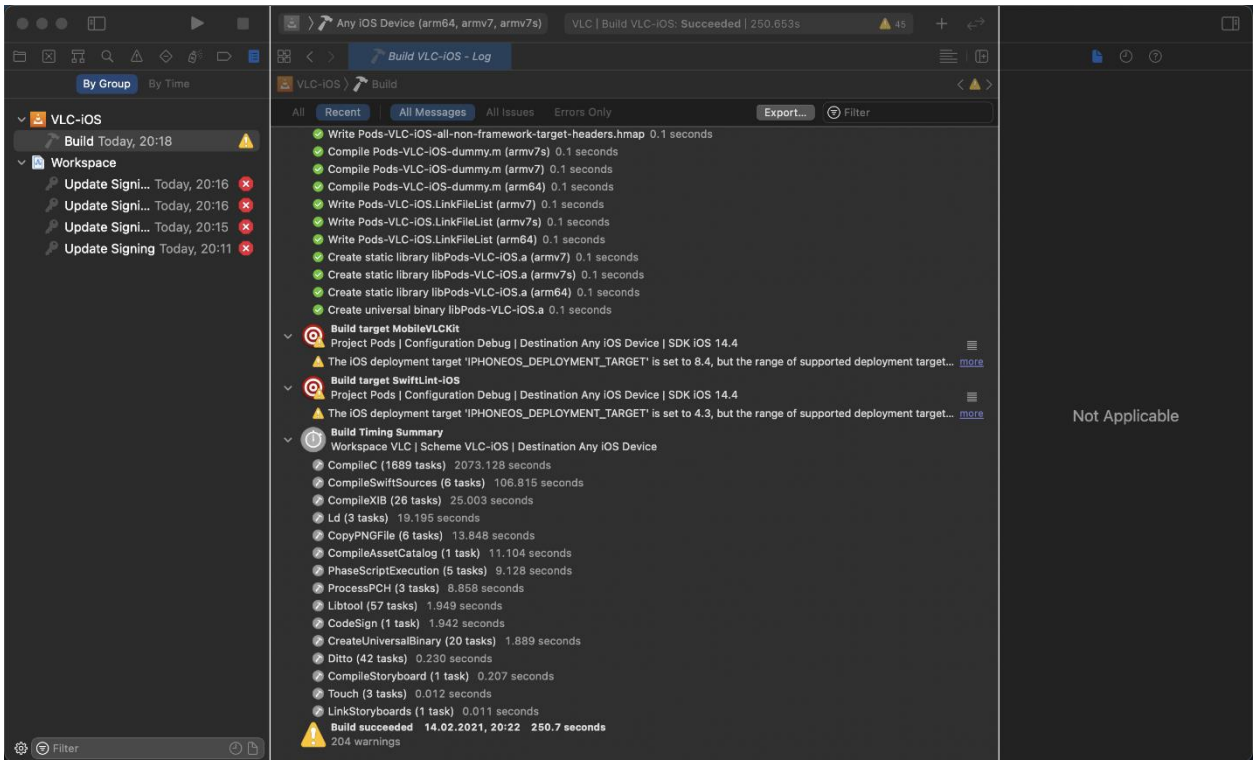
Скриншот 109. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта VLC-iOS в XCode.

M1 справился за 115 секунд, но много это или мало? Сравним.

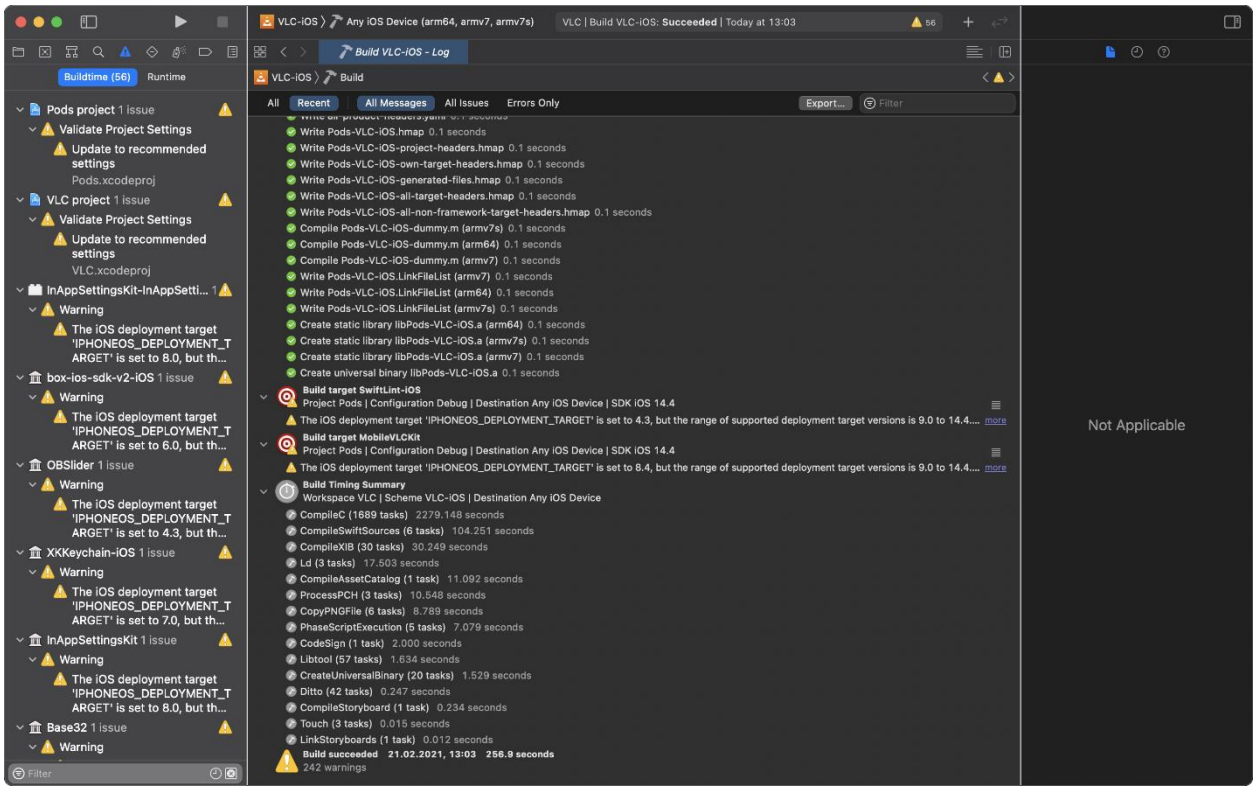


Скриншот 110. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция проекта VLC-iOS в XCode.

Если сравнивать с i7 8550U в Xiaomi, то M1 быстрее в 2.2 раза. Как?!



Скриншот 111. Macbook Pro 13 2018 4x TB3 (i5 8259U, 33W). Компиляция проекта VLC-iOS в XCode.



Скриншот 112. Macbook Pro 13 2020 4x TB3 (i5 1038NG7, 33W). Компиляция проекта VLC-iOS в XCode.

Взгляните также на то, за сколько времени справились Macbook Pro 13 с 4 портами Thunderbolt 3: там разница с Xiaomi в 1% в пользу модели 2018 года и, что странно, в 1% в пользу Xiaomi в сравнении с моделью 2020 года. Ну, это всё укладывается в рамки погрешности, так что особо внимания на это обращать не будем, но информация крайне занимательная.

Действительно Macbook Pro с чипом Apple M1 в 2.2 раза быстрее компилирует Obj-C + Swift проекты под iOS, чем это делают предыдущие поколения Macbook Pro с 4 портами Thunderbolt 3 и чипом Intel Core i5.

А между i5 10-го поколения на 10 нм и i5 8-го поколения на 14 нм я в макбуках какой-то принципиальной разницы не вижу, всё в рамках погрешности в реальной жизни. Гигантский скачок виден только с переходом ноутбуков Apple на собственные чипы Apple Silicon.

Уничтожает ли новый Macbook своих же более дорогих собратьев? Да, выходит, что так, причём как в задачах по рендерингу сцен в Cinebench и Blender, так и в задачах на компиляцию проектов Swift и Obj-C + Swift. Для разработчика, с точки зрения производительности, новые Macbook на M1 будут просто маной небесной. Или нет? Может быть, есть нюансы?

Мне пришла в голову собрать ещё десктопный VLC из исходников, и...



Скриншот 113. Macbook Pro 13 2020 (Apple M1, 20W). Попытка собрать VLC из исходников.

VLC то спокойно собирается на компьютерах с Intel для компьютеров Intel, но не на компьютерах с Apple Silicon, причём сборка там не проходит что под ARM, что под x86 (x86_64). Хотя, пару раз сборка прошла, но это дикая лотерея. Я об этом чуть позднее расскажу.

5.7. Скорость компиляции крупного проекта на Rust.

Один из наших читателей и, по совместительству, разработчик консольной утилиты на Rust для заполнения SSD, Adatan, попросил меня проверить, сколь быстро этот Mac справится с компиляцией крупного проекта на Rust и сравнить его результаты с аналогом на Intel. В качестве аналога выступит мой Xiaomi. По тестам в Cinebench R20 и R23, Geekbench 5 и Blender 2.91.2, думаю, Вы и так поняли, что мой Xiaomi может выступать как аналог Macbook Pro 13 2018 с 4 портами Thunderbolt 3 (i5 8559U), если Вы уже нагуглили результаты мака в сети (этих результатов в сети полно, выбирайте сами, какому изданию больше или меньше доверять), а в XCode между моим Xiaomi и даже относительно свежим Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (i5 1038NG7) никакой существенной разницы не было. Какая же будет разница при сборке проекта на Rust? И будет ли она вообще?

Для теста мы собрали Veloren, мультиплеерный RPG проект.

Для начала установим требуемую рабочую среду Rust

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

После установки для того, чтобы у нас обновились переменные среды ($PATH) без необходимости в перезагрузке, вводим следующую команду:

```
source $HOME/.cargo/env
```

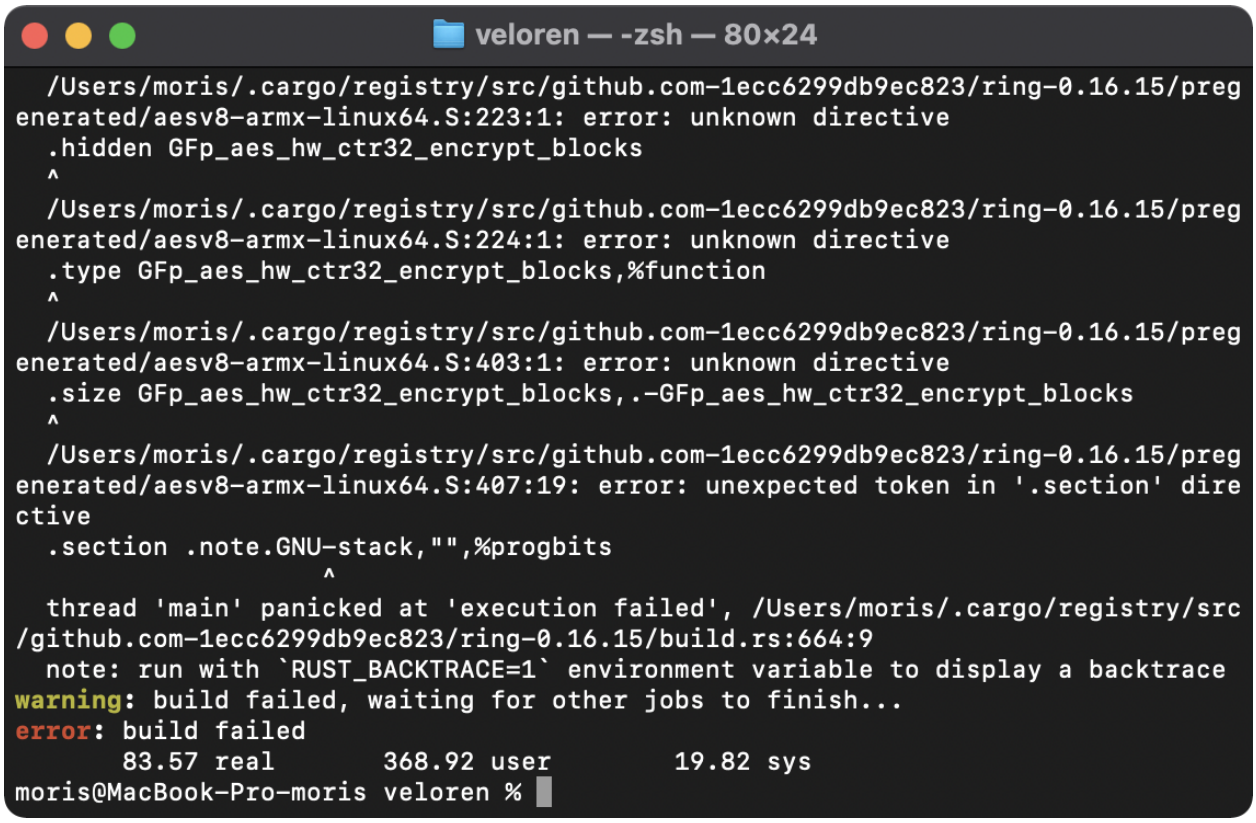
После этого просто копируем к себе проект, приводим его к той же версии, которую мы собирали, а затем собираем. Всё одним набором команд:

```
git clone --recursive https://github.com/veloren/veloren.git; cd veloren; git reset --hard a7ecd8e180b8b5596dddee4dbc605e63eef69669; cargo fetch; rustup target add x86_64-apple-darwin; MACOSX_DEPLOYMENT_TARGET=11.1 OSX_VERSION_MIN=11.1 /usr/bin/time cargo build --release --target x86_64-apple-darwin
```

Для повторной сборки можете задействовать эту команду:

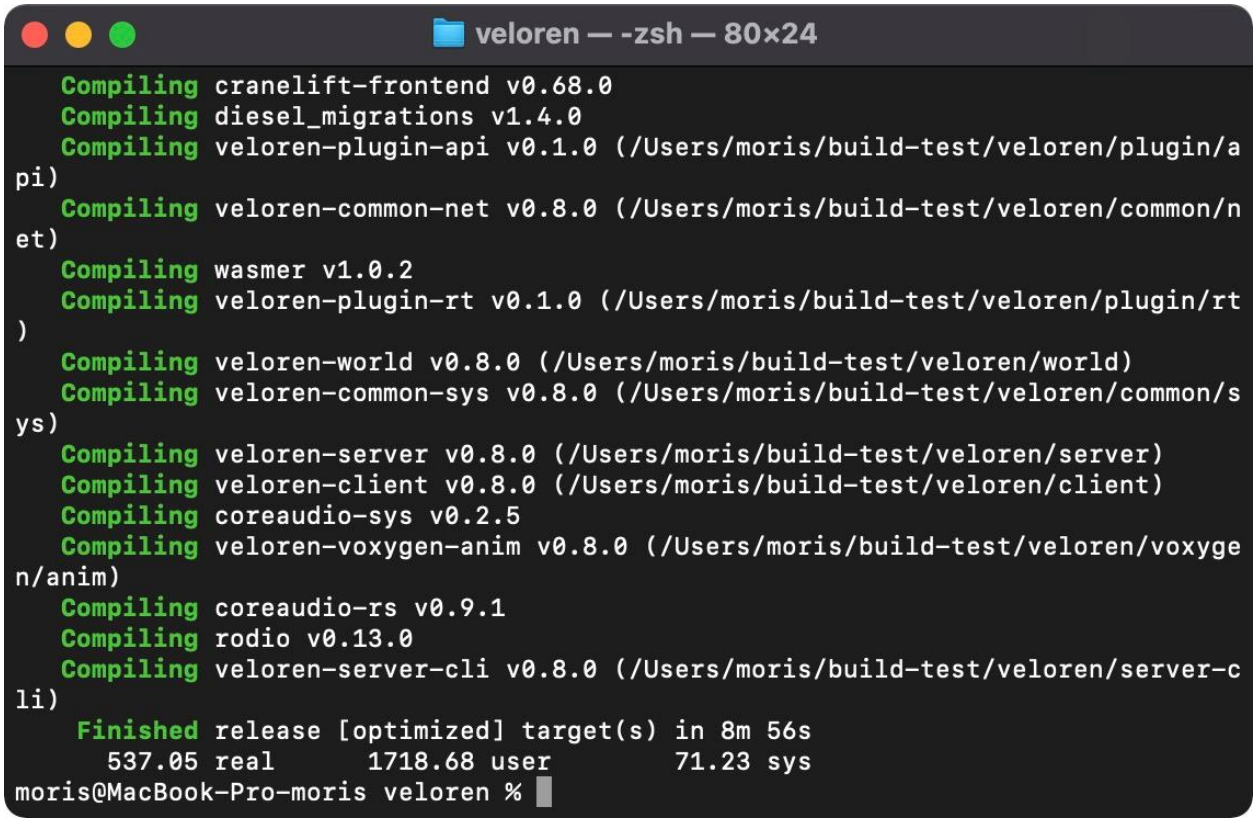
```
cargo clean; rustup target add x86_64-apple-darwin; MACOSX_DEPLOYMENT_TARGET=11.1 OSX_VERSION_MIN=11.1 /usr/bin/time cargo build --release --target x86_64-apple-darwin
```

Собираем мы Veloren под x86_64. Почему не собираем под ARM? Потому, что при сборке под ARM мы ловим ошибку.



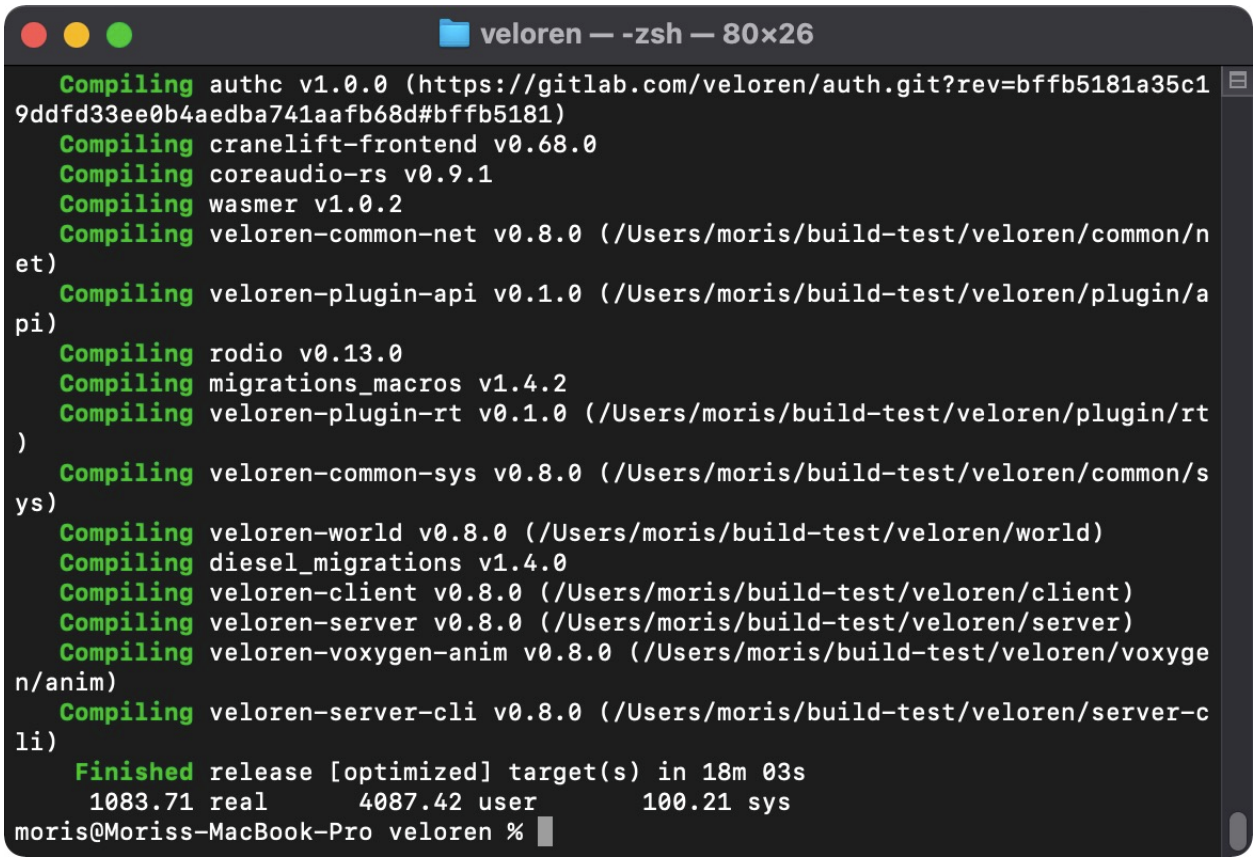
Скриншот 114. Попытка собрать veloren под arm.

Притом, неважно, собираете Вы [veloren] с ARM машины для ARM или с x86_64 машины для ARM. Просто сборка для ARM не пройдёт, тут никак.



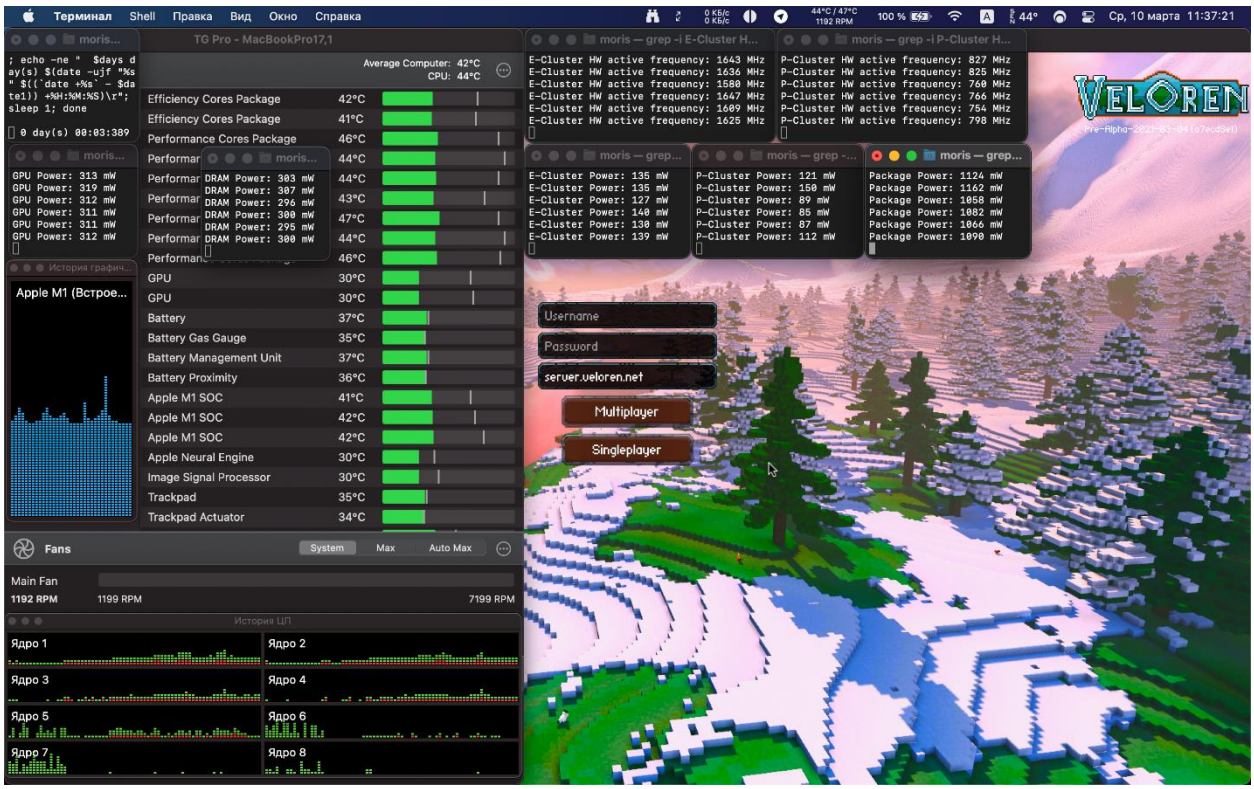
Скриншот 115. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта Veloren на Rust. Коммит от 4 марта 2021.

Macbook Pro на M1 справился за 9 минут. Много это или мало?



Скриншот 116. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция проекта Veloren на Rust. [Git-версия (commit) от 4 марта].

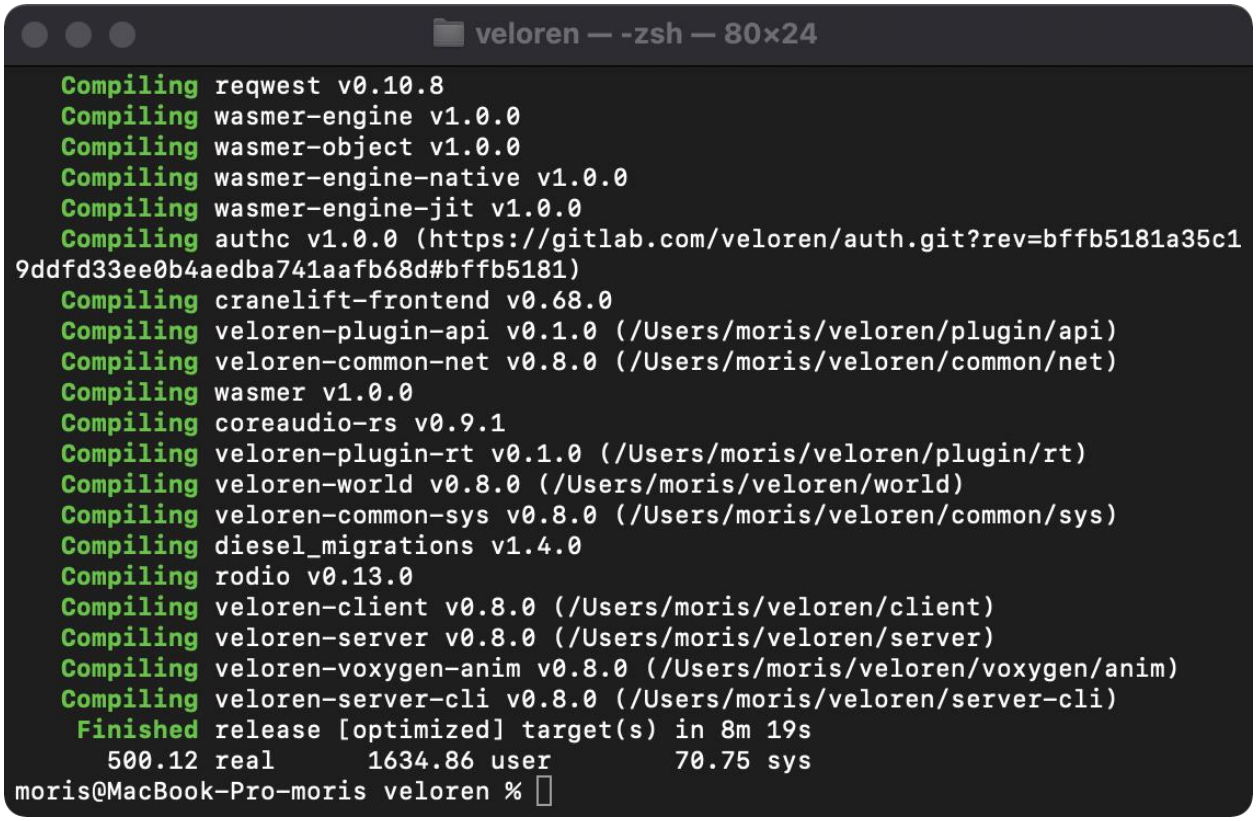
Если сравнить с Xiaomi, разница составляет аж 2,02 раза! Это феноменальный отрыв. И это при том, что мы собирали Veloren исключительно под x86_64, поскольку для ARM его собрать не удаётся.



Скриншот 117. Macbook Pro 13 2020 2x TB3 (Apple M1, 20W). Запуск Veloren через Rosetta 2.

Как видите, Veloren вполне нормально стартует на Macbook с чипом M1, т.к., насколько мне известно (поправьте, если ошибаюсь), у Veloren нет проблем с AVX кодом. Могут возникнуть проблемы с графикой, т.к. Veloren специально под Metal не адаптировали, но в целом возможность собирать проекты в принципе присутствует. Это не лучший вариант в принципе брать для разработки на Rust ноутбук от Apple, тогда как Metal API в первую очередь предназначен для Swift/Obj-C/C++, но в принципе пока OpenGL работает, так можно работать. Но вангую, что в ближайшем будущем OpenGL полностью выпилят из macOS и такого варианта более не будет, и тут уж придётся либо винду юзать (с которой на данный момент имеются проблемы, но об этом в другой главе), либо просто не брать себе Macbook с чипом M1. Такое моё мнение, но моё мнение ни на что не претендует. По факту то всё запускается и стартует. Тут выводы сами делайте.

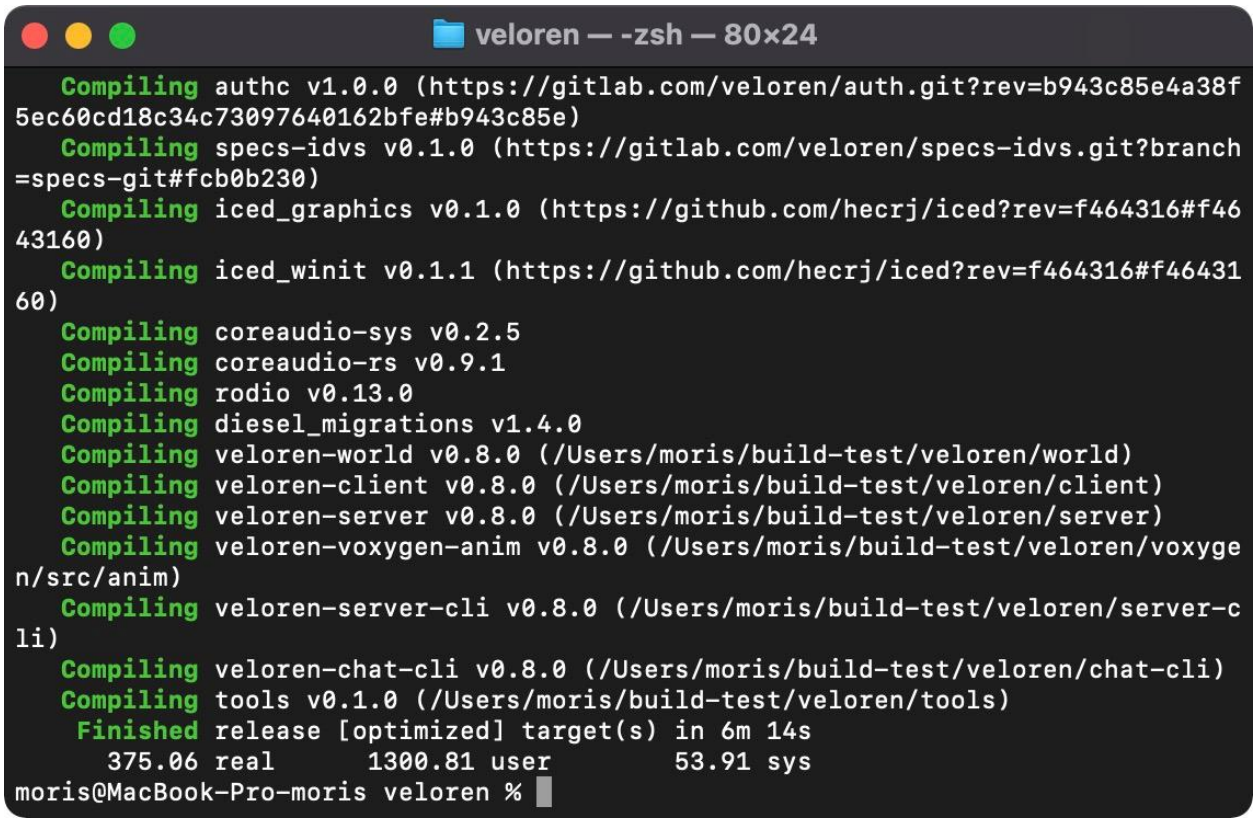
Может ещё возникнуть вопрос, почему я при сборке постоянно использую вначале команду git reset –hard ключ_коммита? Отвечу сразу, чтобы такого вопроса не возникало: у разных проектов разное время сборки, и у разных версий проекта тоже разнится время сборки.



Скриншот 118. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта Veloren на Rust. Git-версия (commit) от 7 февраля 2021.

Если мы попытаемся собрать проект из исходников месячной давности, т.е. если мы «ресетнем» git на 7 февраля, то проект соберётся на M1 уже не за 537 секунд, а за 500 секунд, а на Xiaomi – не за 1083 сек, а за 1035 сек.

Очевидно, со временем проект разрастается, и это сказывается на скорости его компиляции, поэтому я и даю Вам конкретный коммит, к которому стоит вернуться, чтобы сравнить результаты с нашими

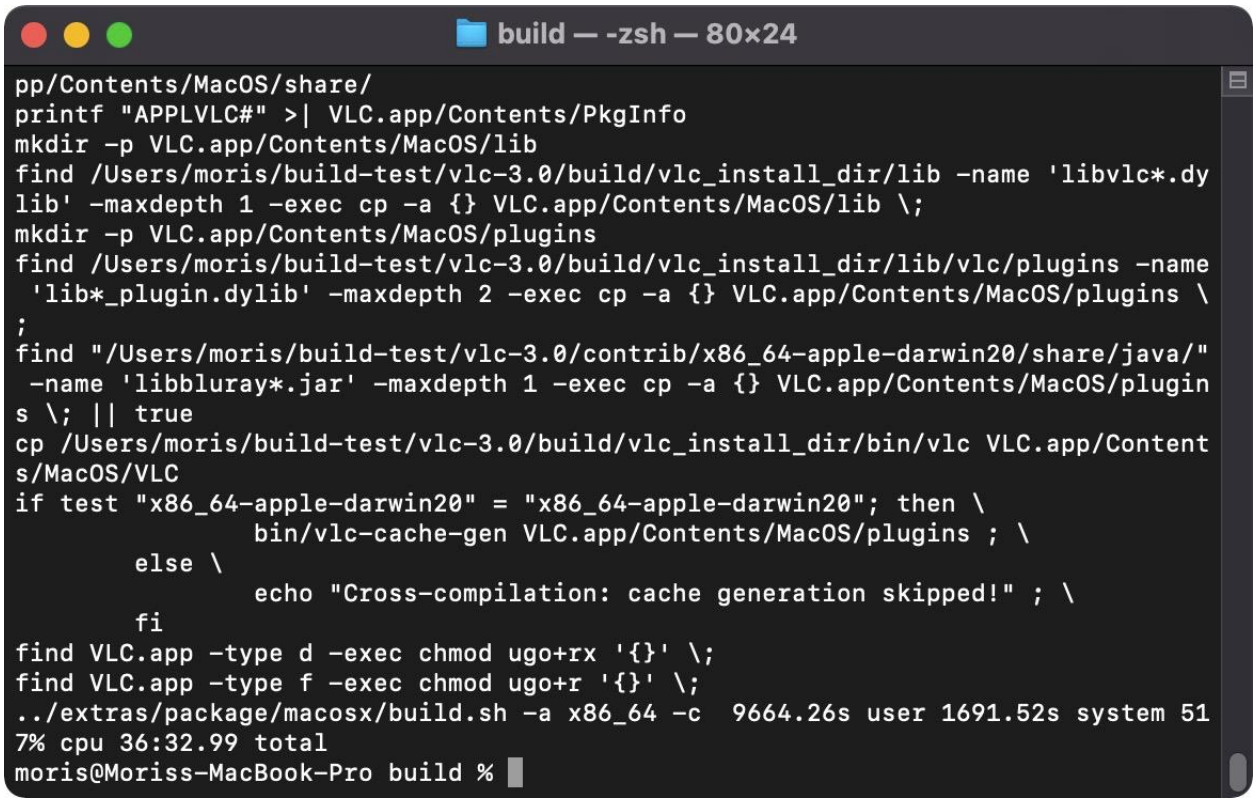


Скриншот 119. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта Veloren (Rust). Версия 0.8 от 28 ноября 2020.

Если совсем «осотонеть» и «ресетнуться» на коммит 4-ёх месячной давности, тем самым откатившись до релизной версии 0.8, время сборки сократится у M1 до 375 сек., а у Xiaomi – до 759 сек. Т.е. разница по-прежнему составляет всё те же 2,02 раза. Прирост просто феноменальный. Да, инструменты Rust (rustup) под macOS с архитектурой ARM выпускаются нативно, начиная с версии 1.49, т.е. для сборки не требуется задействовать Rosetta 2. И всё равно у нас проходит ведь кросс-компиляция под платформу x86_64, т.е. преимущество по идее должно быть у Intel, но нет. По идее Rust не особо годится для разработки под macOS, т.к. API Metal в первую очередь заточено под работу с Swift и Obj-C. Хотя, у Veloren, например, намечается переход на wgpu, который, в свою очередь, транслирует команды в Metal.

5.8. Сборка крупных проектов с множеством компонентов и зависимостей на C, C++, Java и Python.

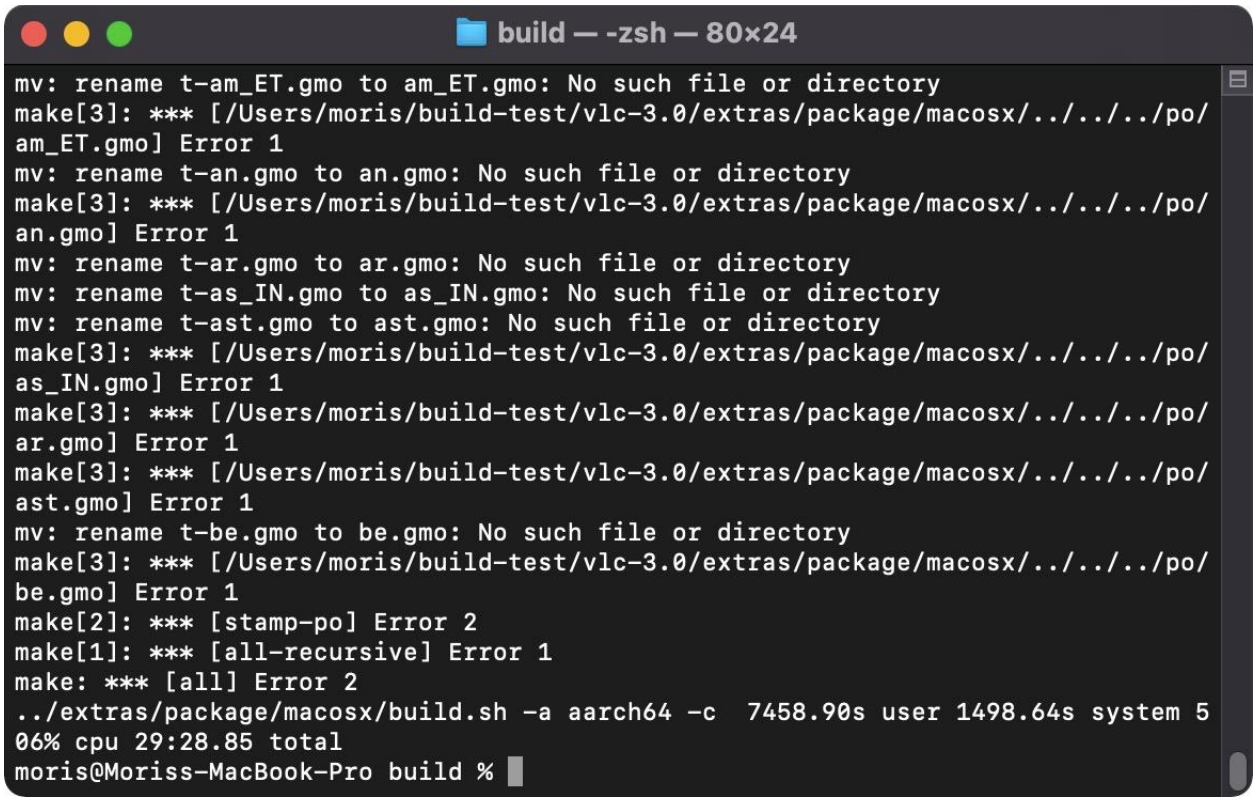
Для сборки я сперва установил Azul (Zulu) сборку OpenJDK 8, python 3.9 через brew, XCode 12.4 (он был установлен ещё при предыдущем тестировании), macOS SDK 11.1 (последняя на данный момент версия), и сборка проводилась под macOS 11.2.1 (11.2 на Xiaomi). Что в итоге?



Скриншот 120. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция проекта VLC-3.0 (тэг 3.0.12.1) под x86_64.

На Xiaomi сборка проходит успешно за 36.5 минут. Уточню, что у меня сборка прошла медленнее, чем прошла бы на MBP 13 2018, т.к. при первой сборке происходит также и загрузка необходимых компонентов из сети, а Wi-Fi модуль в Xiaomi под управлением macOS работает значительно хуже, чем под управлением Windows, т.к. имеющиеся под него драйвера itlwm на данный момент поддерживают лишь ширину беспроводного канала в 20 МГц, тогда как 2.4 ГГц Wi-Fi сеть у меня дома использует ширину в 40 МГц, а 5 ГГц Wi-Fi сеть и вовсе задействует полосу в 80 МГц. Сами понимаете, что Wi-Fi на macOS у меня значительно медленнее, чем на Macbook Pro 13 2018, поэтому сопоставлять эти результаты напрямую с ним и нельзя утверждать, что на MBP 13 2018 с 4 портами Thunderbolt 3 результат был бы схож.

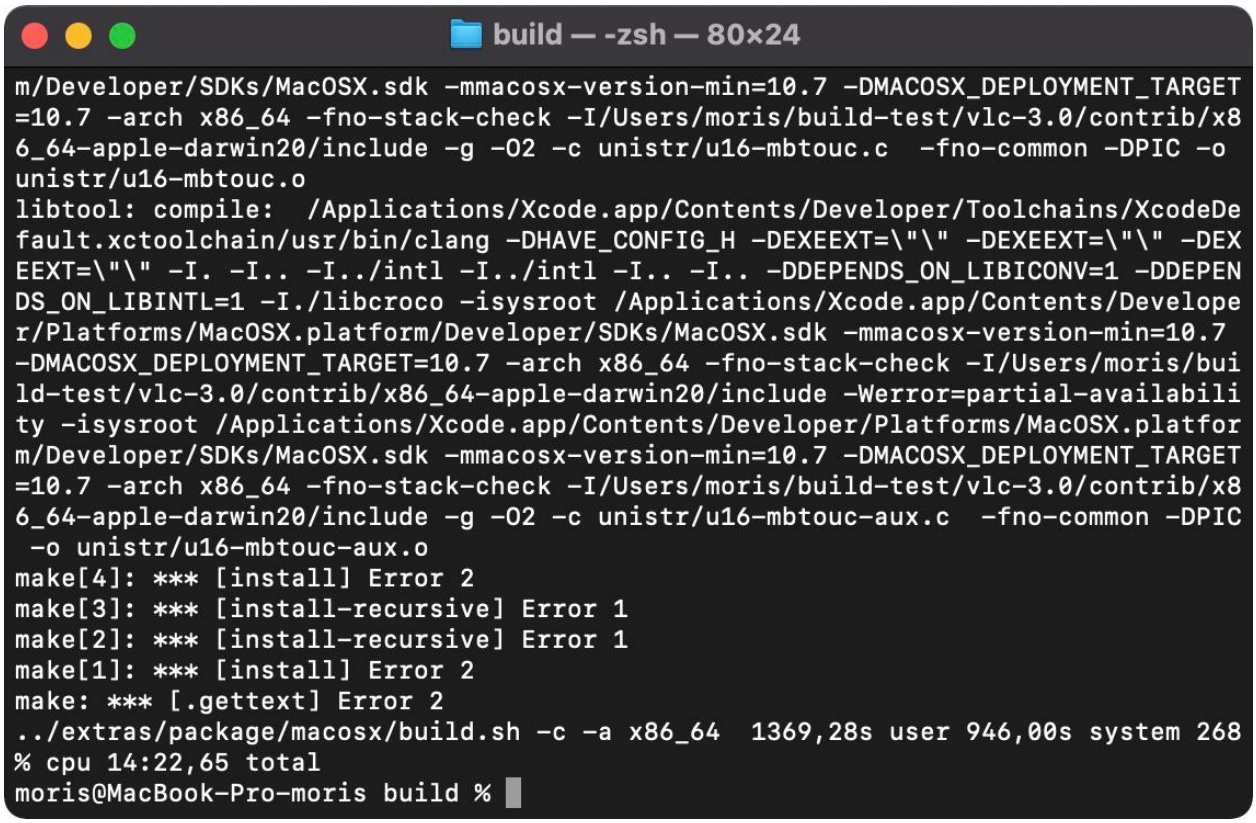
Но важно другое: а VLC вообще собирается? На Xiaomi под x86 – да.



Скриншот 121. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция проекта VLC-3.0 (тэг 3.0.12.1) под ARM.

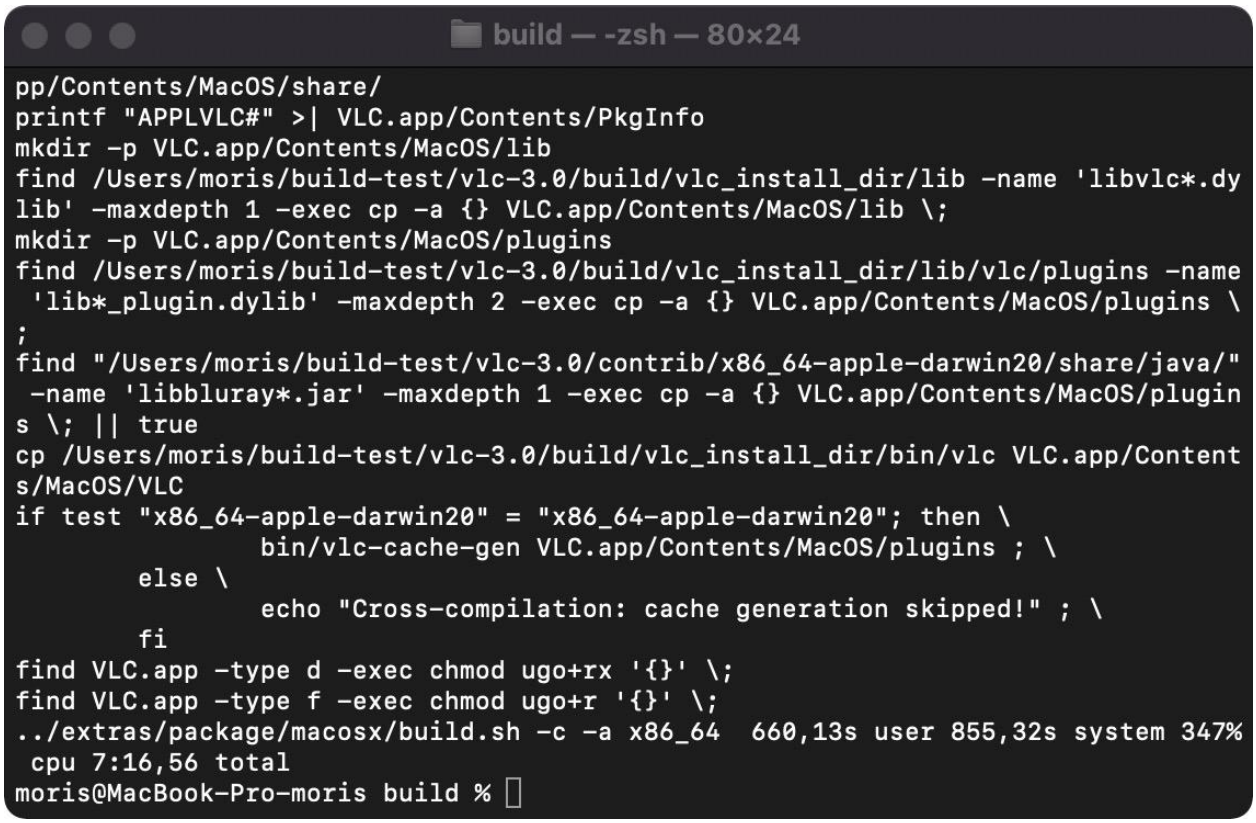
Под ARM собрать его на Xiaomi уже не получается, что бы я ни делал, и это странно, ведь у версии 3.0.12.1 уже заявлена официально поддержка архитектуры ARM на macOS, и есть уже готовые сборки под новые Mac с чипом M1. Но инструкцию сборки для новых маков не обновили, т.е. как его собрать под M1, пришлось додумывать самому. И ничего не вышло.

А что, если проводить сборку на Macbook Pro с чипом M1?



Скриншот 122. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта VLC-3.0 (тэг 3.0.12.1) под x86.

Я потратил около 40 часов своего времени, пытаясь собрать VLC Desktop из исходного кода под macOS с архитектурой ARM (aarch64). Я пробовал короткую инструкцию с одним скриптом (build.sh), пробовал продвинутую инструкцию с Вики, пробовал инструкцию с репозитория, пробовал комбинировать разные решения для сборки, разумеется, гуглил решения ошибок и пытался их применять, я пробовал нативную сборку под ARM, пробовал и сборку под ARM через Rosetta 2, но всё тщетно...



Скриншот 123. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция проекта VLC-3.0 (тэг 3.0.12.1) под x86 через Rosetta.

Мне удалось собрать VLC под x86_64 через Rosetta 2 только 1 раз сразу после перезапуска Terminal через Rosetta 2 и запуска сборки без очистки папки VLC. Но после того, как я выполнил полную очистку всех компонентов VLC, и заново всё загрузил, сборка так успешно и не прошла. В общем, проект то собирается, то нет, никакой определённости. Притом, это я ещё собирал под x86, а под ARM он ни в какую не собирается.

Я не понимаю, как сами разработчики VLC собирают своё детище под macOS машины с архитектурой ARM (aarch64). Я не нашёл у них инструкции по сборке именно версии для ARM, так что этот вопрос остаётся открытом. Геморрой тот ещё и гарантий тут никаких нет.

Вы не можете быть уверены, что, купив Macbook на M1, Вы сможете собрать свой проект, даже запустив Terminal через Rosetta 2, что автоматически заставит все запускаемые процессы через Terminal также стартовать через Rosetta 2. У меня даже так вышло проект собрать лишь 1 раз по чистой случайности, и то на «горячую». На «холодную» сборка ну вот никак не проходит. В общем, если разрабатываете приложения из множества компонентов на C, C++, Java и Python вместе, пока не переходите на M1.

5.9. Скорость компиляции проекта на C + Assembler.

Я много инфы прошерстил про Mac на M1 прежде, чем сесть писать эту статью, и меня привлёк этот пост в Twitter от VideoLAN, разработчиков VLC.

В этом твите сравнивается скорость декодирования видео с кодеком AV1 на Mac Mini с чипом Apple M1 и Macbook Pro 15” или 16” (i9, 9980H тут либо имеется в виду 9880H на 2.3 ГГц, либо 9980HK на 2.4 ГГц, т.к. моделей Macbook Pro с 9980H без K в конце индекса просто нет в природе).

И тут я подумал: а почему бы не собрать этот проект самому и не сравнить скорость сборки, а также скорость декодирования видео AV1? Декодер видео DAV1D не задействует блоки аппаратного декодирования видео AV1 ни в одном чипсете (будь то Intel Core 11-го поколения Tiger Lake на 10 нм, Apple Silicon M1 на 5 нм, или Mediatek Dimensity 1000+ на 7 нм). Т.е. это будет программное декодирование, а не аппаратное, или, проще говоря, декодирование силами ядер центрального процессора, а не каких-либо специально заточенных под это дело вычислительных блоков.

Сперва подготовим необходимое для сборки проекта окружение.

Загружаем XCode через App Store и устанавливаем путь к нему в качестве стандартного пути для источника утилит командной строки XCode:

```
sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer;
```

Установим утилиты XCode для командной строки через Terminal:

```
sudo xcode-select –install
```

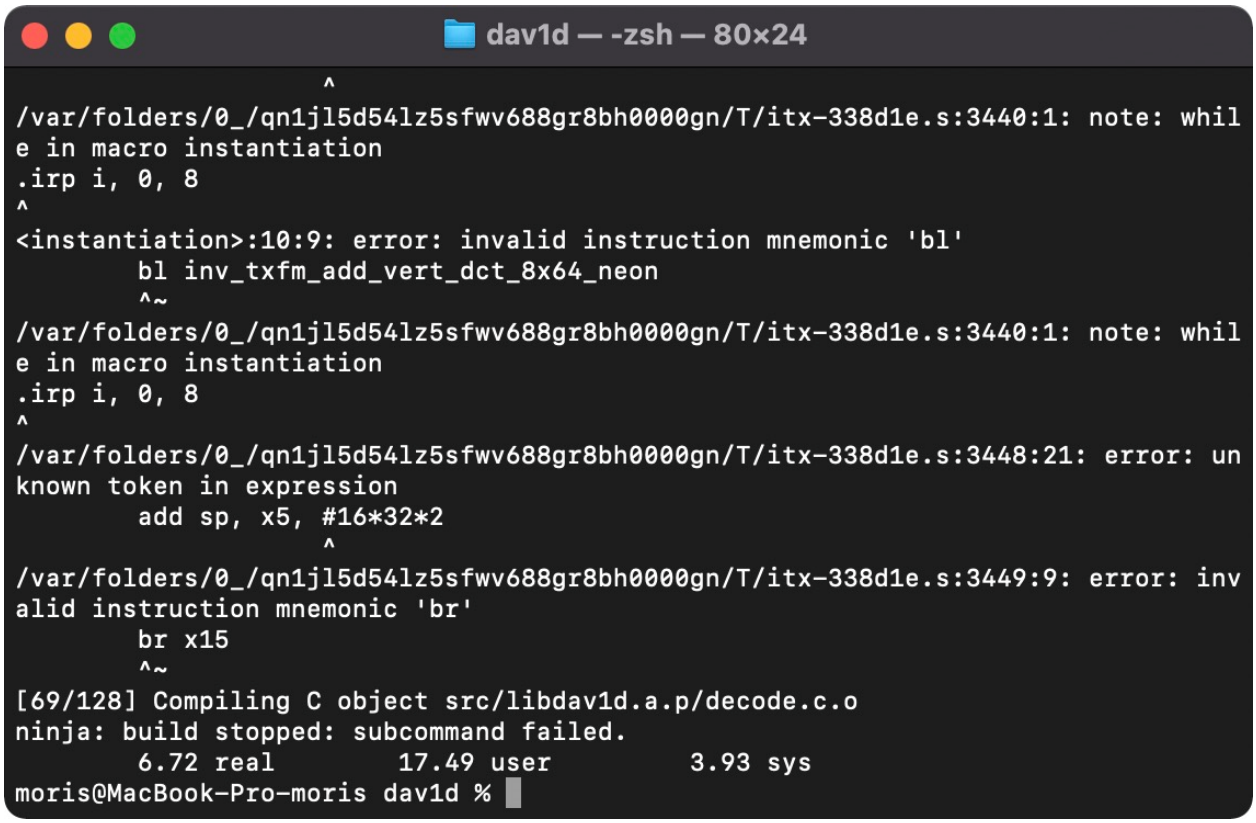
После этого устанавливаем meson и ninja (+ nasm для x86) через brew:

```
brew install meson ninja nasm
```

Тут важное уточнение: для сборки проекта под x86 требуются помимо пакетов Meson и Ninja, а также их зависимостей, ещё и пакет nasm. Для сборки проекта под ARM он не требуется. Ну и кросс-компиляцию в случае с dav1d мы не можем провести как с x86_64 под ARM, так и с ARM под x86.

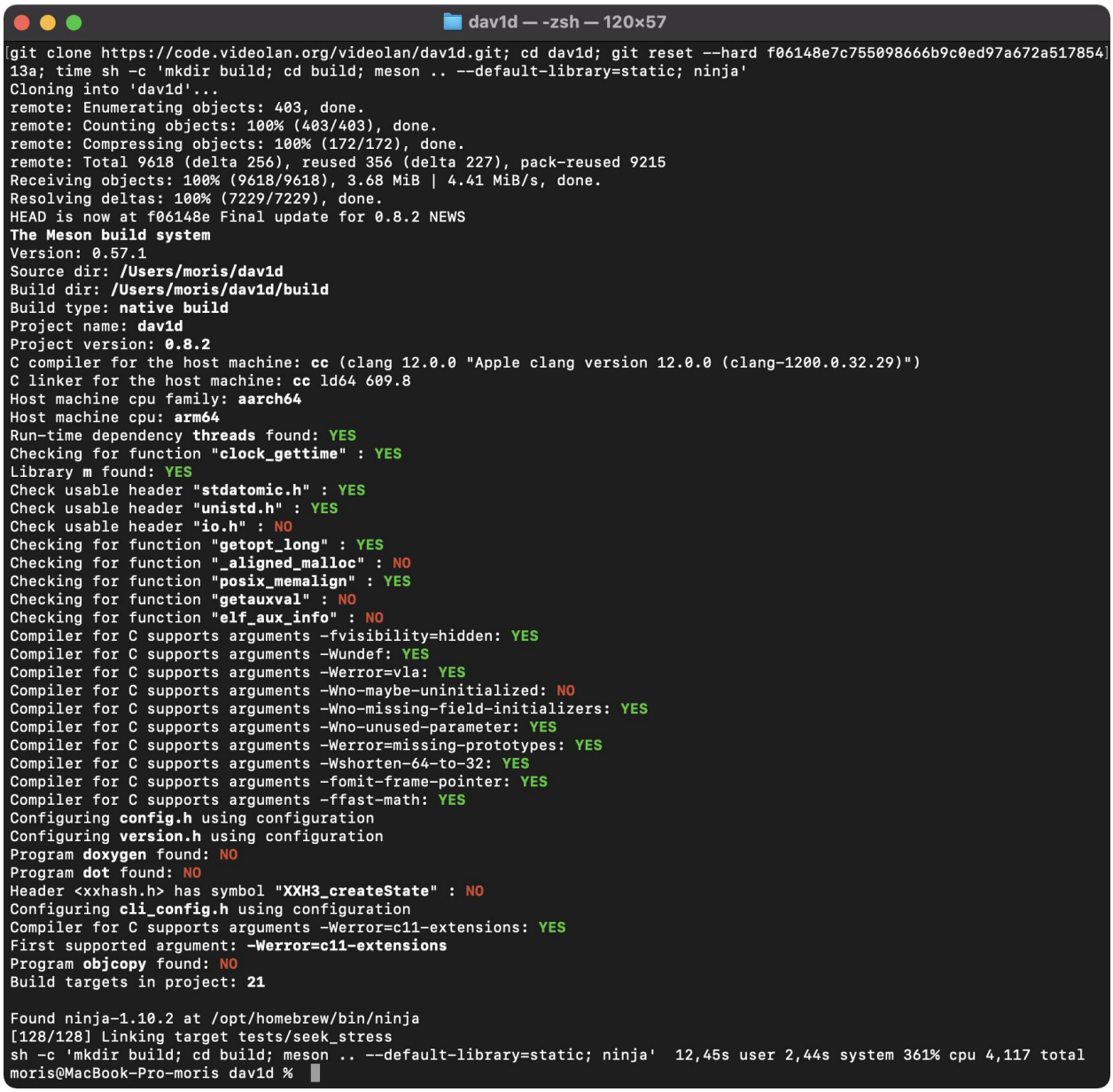
Набором команд ниже мы загрузим исходный код dav1d и соберём его.

```
git clone https://code.videolan.org/videolan/dav1d.git; cd dav1d; git reset --
hard f06148e7c755098666b9c0ed97a672a51785413a; time sh -c 'mkdir build; cd build;
meson .. --default-library=static; ninja'
```



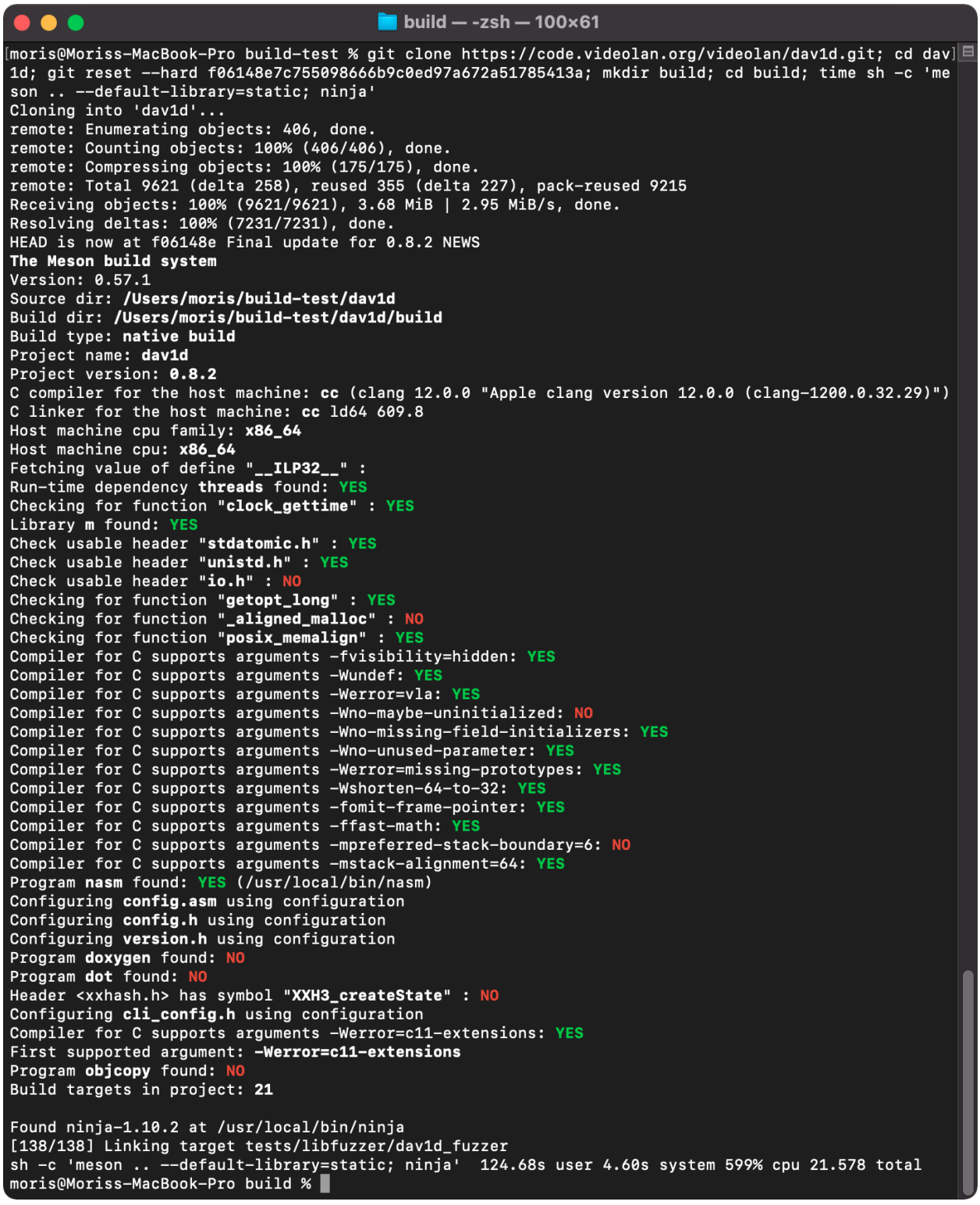
Скриншот 124. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция dav1d. Ошибка при использовании Rosetta 2.

Из-за ошибки при сборке из-под Rosetta и отсутствию нужных файлов для кросс-компиляции, выходит, что с каждой платформы мы можем собрать dav1d только под неё же, и при этом для сборки требуются ещё и разные компоненты. Максимально странный тест, казалось бы, ведь мало того, что для сборки требуются разные рабочие окружения, так ещё и в составе dav1d присутствует масса кода на Ассемблере. Как Вы все знаете, Ассемблерный код – по сути и есть машинный код, только представленный в более-менее понятном виде. Ассемблерный код в составе dav1d для aarch64 (ARM64) и x86_64 (x86) разный, ведь и наборы команд они поддерживают разные). Т.е. у нас не только разные окружения требуются для сборки, но и собирается разный код. Учитывая то, что Apple уже начала отказываться от SoC Intel, и новых компьютеров с SoC Intel под macOS ждать особо не стоит, выходит, что этот тест более-менее отражает разницу между сборкой под C+Asm под x86_64 в недалёком прошлом и тем, что мы с Вами будем наблюдать в будущем, когда сложное низкоуровневое ПО будет собираться только под Apple Silicon.



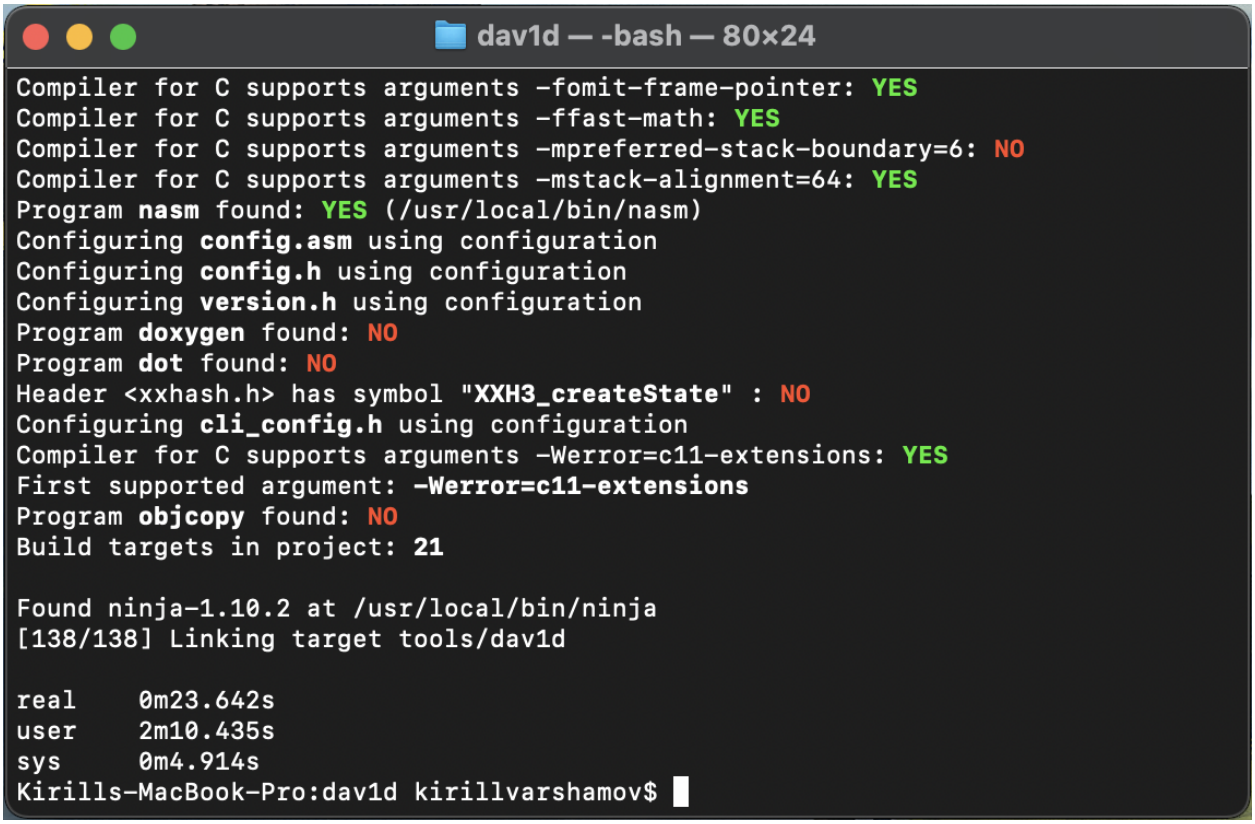
Скриншот 125. Macbook Pro 13 2020 (Apple M1, 20W). Компиляция dav1d. Скорость сборки под Apple Silicon.

Macbook Pro с чипом Apple M1 справился за 4 секунды. Поясню: 4 секунды – это общее время, затраченное как на конфигурацию проекта через meson (я конфигурировал проект под статическое связывание библиотек), и сборку через ninja. А за сколько справился Xiaomi?



Скриншот 126. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция dav1d. Скорость сборки под Intel.

У Xiaomi на компиляцию, при которой также был задействован модуль nasm (для компиляции под x86_64 он обязателен в отличие от компиляции под aarch64), т.е. задача отличается, хотя это и так было понятно, ведь и платформы разные, так что и задача по компиляции тут разная. Но что в итоге то? 21.5 секунды против 4.1 секунды. Разница в 5,24 раза. Может быть, у меня просто Xiaomi тупит со своим хакинтошем?



Скриншот 127. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Компиляция dav1d. Скорость сборки под Intel.

Я попросил своего друга, Кирилла (kavars), провести у себя сборку dav1d на Macbook Pro 13 2018 4x TB3 (i5 8259U) и у него результат вышел ещё плачевнее: 23.5 секунды вместо моих 21.5 секунды. Разница в целом не критичная, но сколько бы мы ни повторяли тест, у меня результат составлял 21.5 секунды, а у него 23.5 секунды, т.е. это именно не погрешность, а закономерность в ходе тестирования. Впрочем, вполне возможно, что у него просто Операционная Система в фоне стала грузить процессор синхронизацией с iCloud или любой другой фоновой активностью, и разница обусловлена этим, так что ничего утверждать наверняка не буду.

Главное: новый Macbook Pro на M1 код низкого уровня под себя же компилирует в 5 раз быстрее, чем под себя же это делают Macbook Pro с 4 портами Thunderbolt 3 предыдущих поколений. Это огромная разница, и да, она обусловлена в том числе и разными требованиями к рабочему окружению (nasm) для компиляции, и да, там очень много нюансов, но в целом, если смотреть именно на задачу сборки на языке низкого уровня, в будущем компьютеры, которые будут собирать только под M1, будут это делать в 5 раз быстрее, чем ранее выполняли ту же задачу Mac на Intel.

5.10. Сравнительная таблица с результатами тестирования процессора и выводы.

В сравнительную я данные по скорости компиляции заносить не буду, потому что мы не будем так тестировать каждый приехавший к нам ноутбук. Начнём с того, что инструменты для разработки на Windows другие, они могут иметь разные зависимости и сборка с Windows под Windows будет тем более отличаться от сборки с macOS под macOS, т.е. это будет разная нагрузка на железо и, соответственно, разная скорость сборки будет на разных ОС. Но дело не только в этом. Я просто не выдержу так каждый ноутбук тестировать. Я более 2 месяцев потратил на тесты этого ноутбука, да и даже пока я пишу эту статью, я всё ещё провожу некоторые тесты, которые ранее не учёл, но в процессе написания, счёл нужными отразить в статье. И я со своими экспериментами за последние 2 месяца не спал около 20 ночей. Может быть, пару-тройку тестов по компиляции мы начнём проводить, но только если кто-то мне в деталях распишет, как собирать указанные проекты на Windows, чтобы я не сидел как с VLC и не тратил свои бессонные ночи на танцы с бубном, а имел чёткий инструктаж с чётко заранее известным фактом достижения успеха при выполнении указанных действий. Просто «а собери это, или собери вот то, но инструкцию я сам не проверял, вот тут по ссылке читай, авось сработает, а авось – нет», тут уж нет, извините. Я могу просто разово проверить, проходит ли сборка в принципе по инструкции, но, если не проходит, я не буду тратить свои время и нервы на решение проблем со сборкой. Вот я сейчас набил себе шишек и примерно понимаю, что и как собирать, когда в следующий раз мне приедет ноутбук от Apple, так что в следующий раз это месяца 2 не займёт, да и ноутбуки на macOS мне приезжают крайне редко (это первый подобный случай), так что можно иногда и выделить на это больше времени, чем обычно.

Повторюсь: расписывайте мне инструкцию по сборке толковых проектов под Windows с открытым исходным кодом, я протестирую. Но только инструкция должна быть проверена, и все нюансы в ней должны быть уже заранее учтены, а не чтобы я в лотерею играл «соберётся или нет».

Вновь выражаю огромную благодарность Кириллу Варшамову (kavars), Владимиру Барусу, а также Adatan`у за помощь с тестами, в т.ч. компиляции.

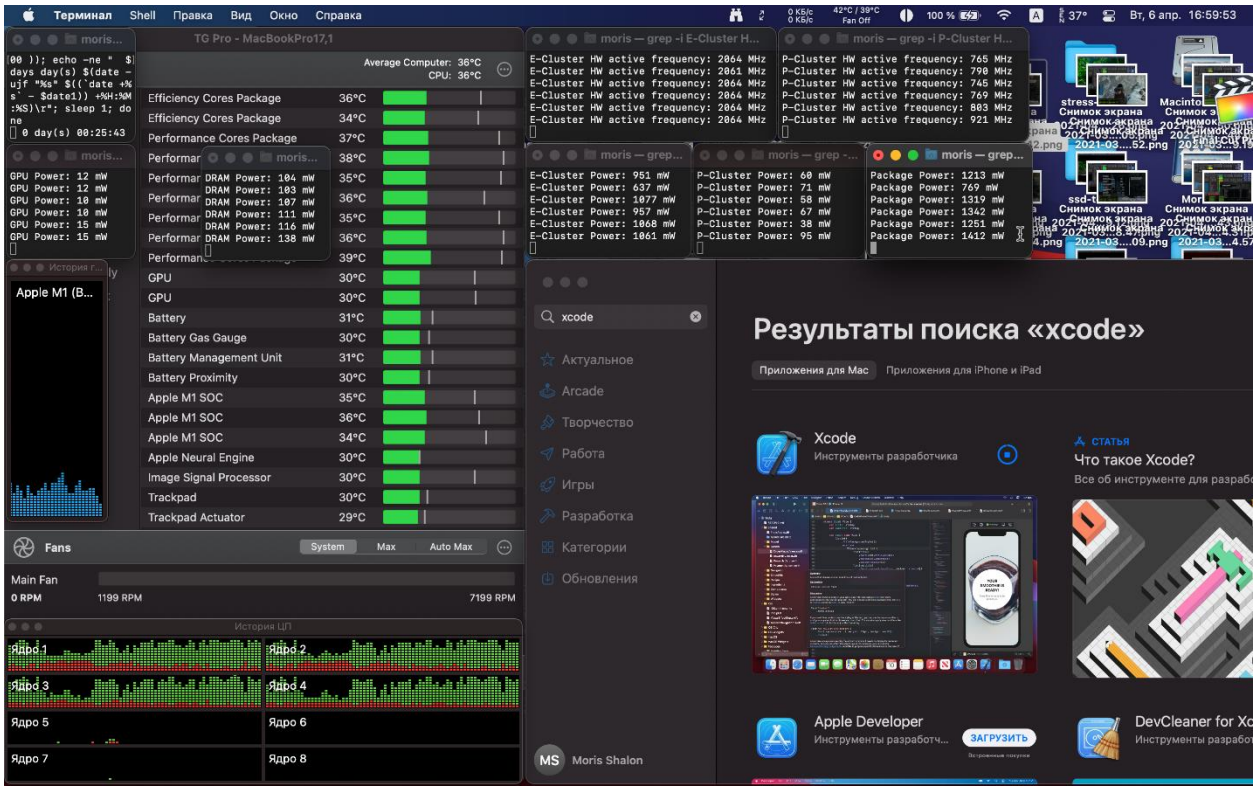
А вот, собственно, и сама сравнительная таблица с выводами по результатам тестирования (вкл. выводы по тому, что в таблицу не заносим):

		Xiaomi Mi Notebook Pro GTX (Core i7 8550U) macOS 25W	Asus ZenBook Flip S UX371E (Core i7 1165G7) 15W	Asus ROG Strix G15 (Ryzen 9 5900HX) 80W	Lenovo Yoga Slim 7 (Ryzen 7 4800U) Extreme Performance 25W	Lenovo Yoga Slim 7 (Ryzen 7 4800U) Intelligent Cooling 20W	Apple Macbook Pro 13 2020 Z11B0004T (Apple M1) Native 20W	Apple Macbook Pro 13 2020 Z11B0004T (Apple M1) Rosetta 2 20W
Cinebench R20	Multi Core	1633	1758	5170	3653	3578		2099
	Single Core	390	439	578	472	472		400
	MP Ratio	4,19	4,01	8,94	7,74	7,58		5,25
Cinebench R23	Multi Core	4221	3994	13359	9317	9132	7807	5184
	Single Core	989	1356	1487	1221	1211	1503	987
	MP Ratio	4,27	2,95	8,98	7,63	7,54	5,19	5,25
Geekbench 5	Multi Core	3823		7422	7028	6923	7669	6051
	Single Core	1001		1499	1146	1148	1746	1340
Blender 2.90 (seconds)	BMW27	486	628	191	258	326		
	Classroom	1348	1808	505	738	896		
	fishy_cat	667	875	252	369	447		
	koro	631	1270	371	555	703		
	pavillon_barcelona	1454	1829	522	786	947		
	victor	2317	3037	883	1420	1647		
Blender 2.91.2 (seconds)	BMW27	484		192	262	308		330
	Classroom	1324		507	748	900		936
	fishy_cat	657		251	371	446		485
	koro	634		371	562	699		576
	pavillon_barcelona	1436		524	798	938		994
	victor	2285		879	1426	1646		1734
Blender 2.92 (seconds)	BMW27	491		194	261	322		330
	Classroom	1369		502	744	901		934
	fishy_cat	664		250	369	447		488
	koro	649		371	561	701		602
	pavillon_barcelona	1467		521	789	948		996
	victor	2300		871	1414	1641		1751

Таблица 2. Сравнительная таблица с результатами тестирования процессоров в ноутбуках.

Вы наверняка смотрите на эту таблицу и восхищаетесь внушительной производительностью Apple M1 при его скромном тепловыделении.

Но прежде, чем мы перейдём к выводам, остался 1 вопрос, который следует затронуть. Если же Macbook Pro с Apple M1 такой быстрый, то почему же он устанавливает приложения из AppStore по часу и более?



Скриншот 128. Мониторинг повторной установки XCode через AppStore после удаления.

В своём ролике «🏆Ты ОФИГЕЕШЬ!🤯MacBook Air M1 vs MacBook Pro M1 vs MacBook Pro 16 (intel)🤯» блоггер Роман Ястреб, видеоролики которого я периодически смотрю, и которого я упоминал во 2-ой главе, на 31:51 рассказал о проблеме очень долгой установки и обновления приложений через AppStore. И я нашёл этому объяснение.

Взгляните на мониторинг на скриншоте выше. Во время установки приложений из AppStore практически вся нагрузка ложится на маломощные ядра процессора, которые в сумме потребляют около 1 Ватта энергии. Задумайтесь сами: может ли блок ядер процессора с со столь низким энергопотреблением быть производительным? По мне, Apple всё правильно сделали, автономность при многозадачности куда важнее скорости установки приложений, но из-за этого приложения и устанавливаются по итогу долго.

Выводы по производительности процессора: это разнос. Менее ли эффективна работа приложений через Rosetta 2 в сравнении с их нативным исполнением, если они сразу из исходного кода собраны под aarch64 (ARM)? Да, Вы видели, что одно и то же приложение (Cinebench R23) с Rosetta 2 может исполняться в 1.5 раза медленнее (или в 1.3 раза в случае с GeekBench

5). Да, трансляция самих приложений с одной архитектуры в другую перед исполнением намного эффективнее трансляции «на лету», и тем более эффективнее эмуляции, но это далеко не то же самое, что работа приложения, сразу из исходного кода, скомпилированного под Вашу архитектуру. И даже так Macbook Pro 13 с чипом Apple M1 демонстрирует выдающиеся результаты и даже не уступает своим более дорогим аналогам от Apple с 4 портами Thunderbolt 3 и чипом Intel Core i5 (будь то 8259U в модели 2018 года или 1038NG7 в модели 2020 года). По тестам скорости компиляции Macbook Pro с чипом M1 лидировал в 2 и более раза в тестах XCode (что при компиляции под Mac на Intel и Mac на Apple Silicon, что при компиляции под iOS), в 2 раза быстрее компилируются такие тяжёлые проекты на Rust как Veloren (при том, что компиляция проходила под x86 платформу, т.к. под ARM скомпилировать не удалось), в 5 раз быстрее M1 компилирует C (+ Ассемблер) код под Apple Silicon, нежели Mac на Intel собирает его под себя же. С точки зрения разработчика, наверное, M1 это манна небесная, но стоит вспомнить про то, с каким геморроем я столкнулся при попытке скомпилировать на M1 проект VLC из исходного кода (версия 3.0.12.1, которая поддерживает Apple Silicon), как все эти мысли улетучиваются. У Вас, возможно, будет работать вся Ваша рабочая среда под M1 нативно или, хотя бы, через Rosetta 2, и тогда есть смысл покупать Mac на M1. Но вполне может быть и так, что в зависимостях Вашего проекта используется код, который даже Rosetta 2 не способна транслировать (например, AVX-инструкции), в таком случае ни за что не берите Mac на M1.

Стоит ли чего-то подобного ждать на Windows? Думаю, что нет. Как-никак Apple лицензирует только архитектуру, а ядра (производительные и энергоэффективные) она разрабатывает сама. Я в главе 5.3 показывал, что аналогов по производительности у Apple M1 нет. Столь мощный чип - чисто заслуга Apple, а не ARM. У других производителей ничего подобного нет и не будет, т.к. они лицензируют готовые ядра Cortex-A7X и Cortex-X1, которые в 1.5-2.5 медленнее. В добавок, Windows то использует эмуляцию, а не трансляцию, в отличие от Rosetta 2 в macOS, что вообще не эффективно....

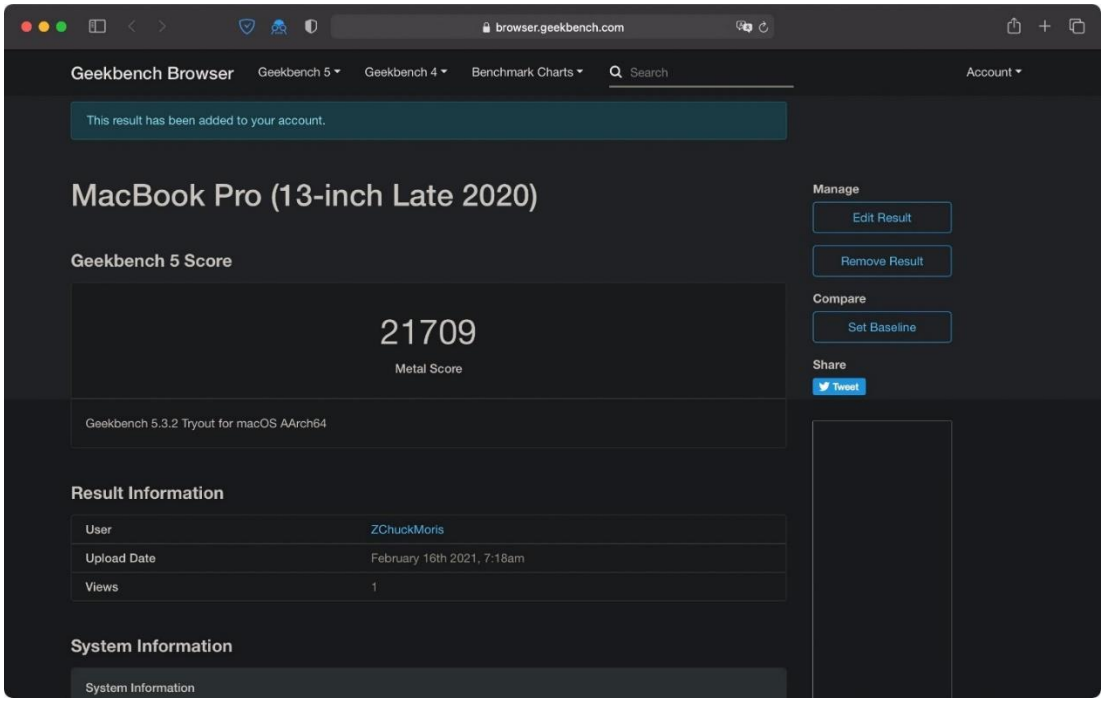
6. Тесты производительности интегрированного графического ускорителя.

В главе 2.8 я подробно расписал, почему сейчас отсутствует поддержка внешних видеокарт от AMD и NVidia, и, скорее всего, нам и не следует ждать их поддержки внешних видеокарт в будущем. Хотя, учитывая то, что Вы увидите далее, вполне может быть и так, что Apple представит свою собственную разработку, которая будет намного лучше решений от конкурентов. И главным её достоинством будет минимальное энергопотребление при сопоставимой производительности. Помните троттлинг-тест GPU, который мы провели при помощи панели настроек в игре Tomb Raider (2013) со средними настройками графики? Также мы мониторили и потребление энергии интегрированным графическим ускорителем во время тестирования сцены Aztec Ruins (Offscreen) в GFXBench Metal? Это позволило нам узнать, что интегрированная графика в Apple M1 потребляет в пике всего 10.4 Ватта в пике. Это по 1.3 Ватта на графический блок. 8 графических блоков, 10.4 Ватта потребления энергии. Разве это может нас хоть как-либо впечатлить? Разве может такая графика быть производительной? У нас на тесты бывали ноутбуки с видеокартами, которые потребляли до 150 Ватт энергии и выделяли невообразимое количество тепла. Разве нас может удивить видеокарта, у которой графика потребляет столь мало энергии?

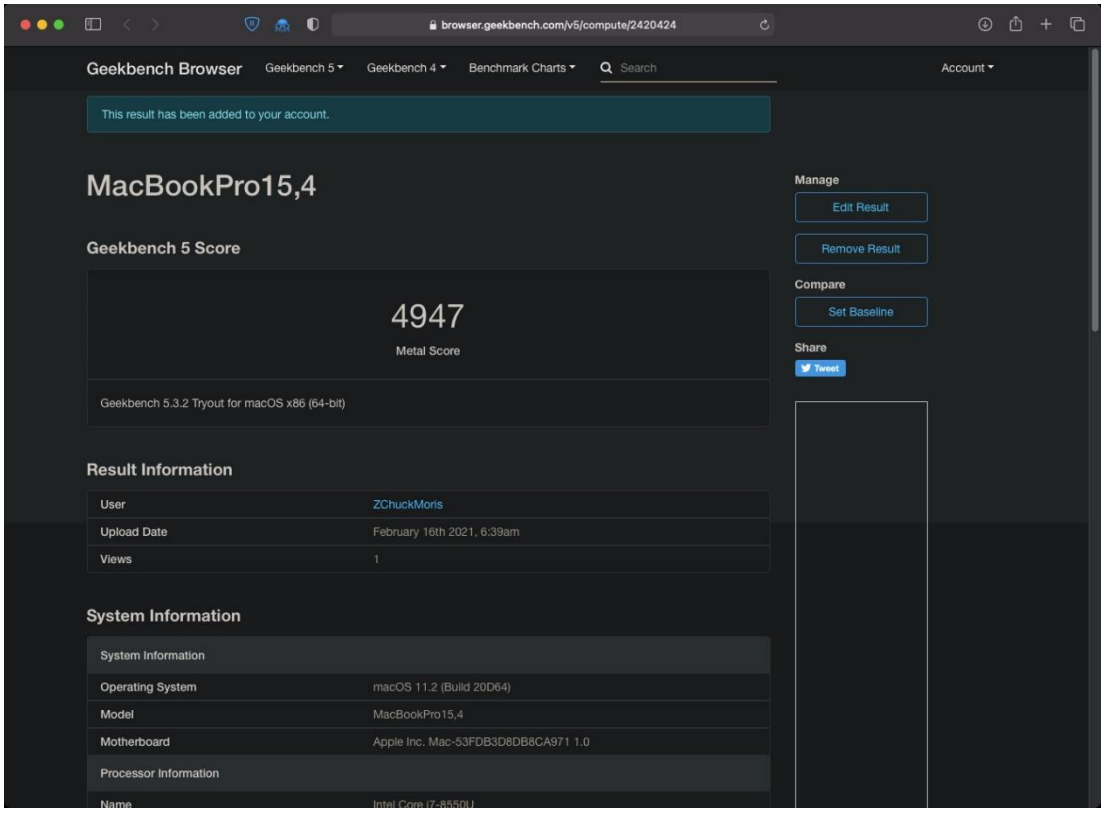
Спойлер: да, она ещё как меня удивила, но с видеокартами, которые потребляют 50+ Вт энергии она, разумеется, не сравнится. Если выйдет графика от Apple с 64 графическими блоками, будьте уверены, она даст «лососнуть тунца» любой ноутбучной графике, которая существует на сегодняшний день. С нетерпением этого жду. А почему я этого жду, Вы узнаете ниже.

6.1. Geekbench 5 Compute (API OpenCL, Metal).

Начнём с тестов, результаты которых Вы и так можете найти в сети.



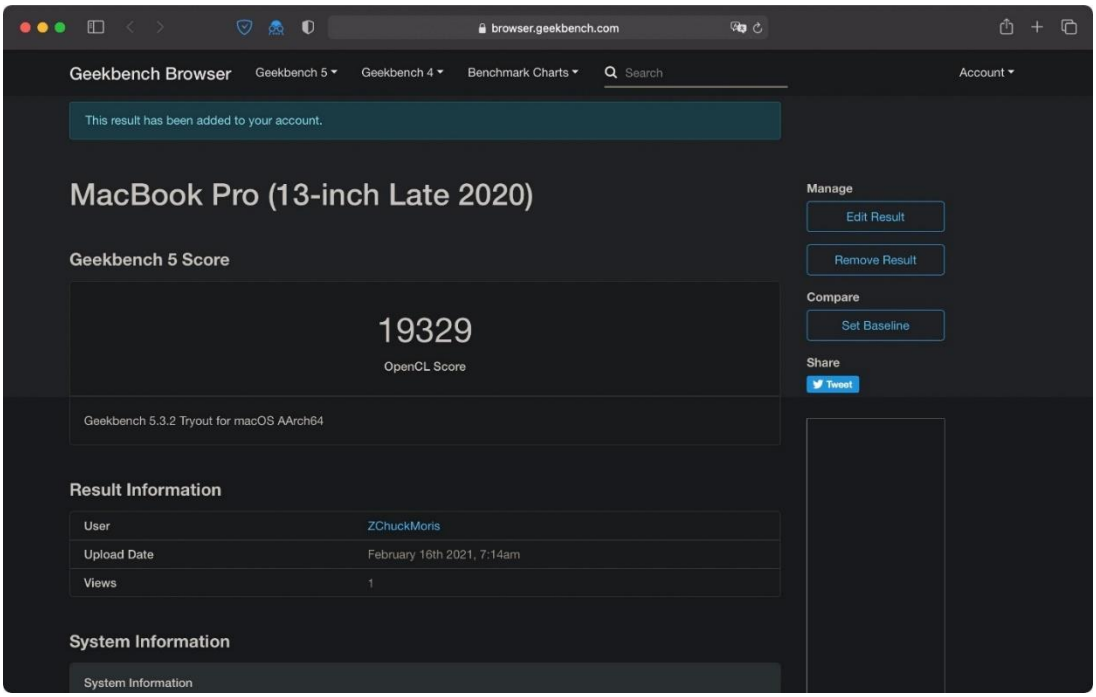
Скриншот 129. Macbook Pro 13 2020 (Apple M1, 20W). Geekbench 5. Compute Test. API Metal.



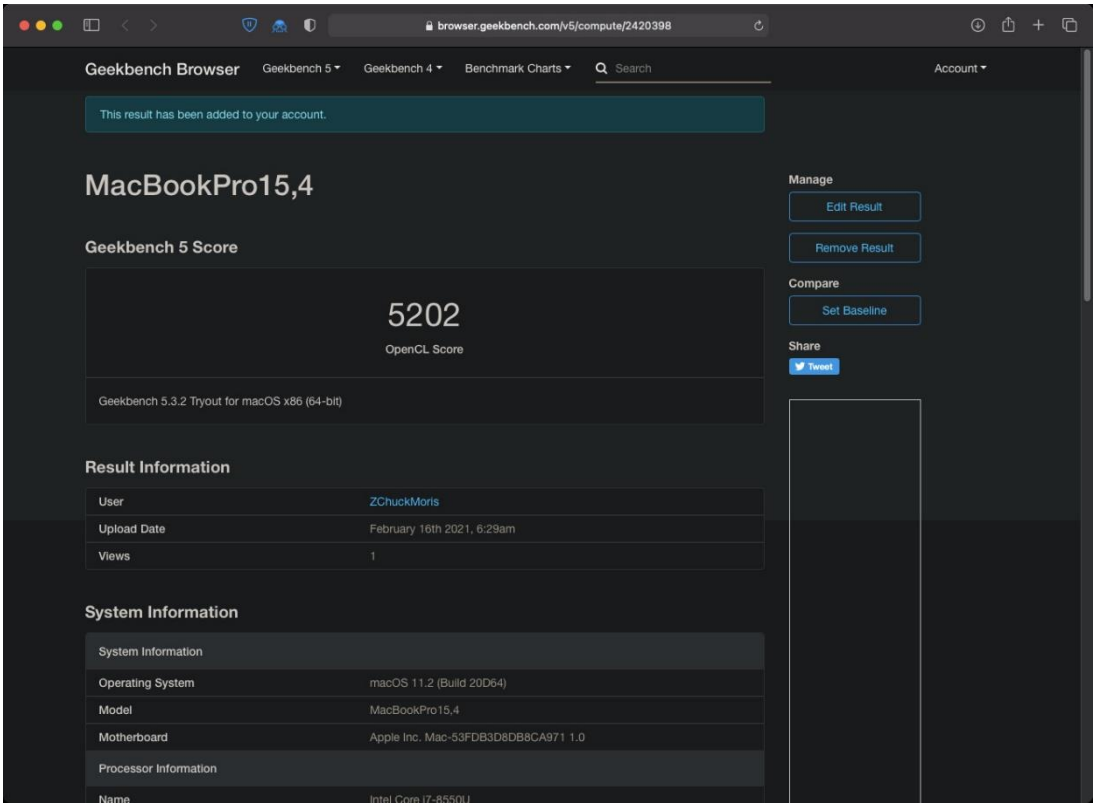
Скриншот 130. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Geekbench 5. Compute Test. API Metal.

В Geekbench 5 новый Macbook Pro с SoC Apple M1 при обсчёте на видеоускорителе с помощью API Metal набирает в 4.38 раза больше баллов и в 3 раза больше баллов, чем Macbook Pro 2018 с 4x TB3 (i5 8259U) или

Macbook Pro 2019 с 2x TB3 (i5 8257U), что видно в базе с результатами Geekbench 5. Если сравнивать с Macbook Pro 2020 с 4x TB3 (i5 1038NG7), то разница с ним составляет уже 2,08 раза. Да, Intel в 10-ом поколении своих SoC на базе 10 нм техпроцесса (и далее в 11-ом поколении) совершили большой скачок вперёд по части графики, но это далеко не уровень Apple M1. Это всё актуально, если сравнивать результаты с API Metal.



Скриншот 131. Macbook Pro 13 2020 (Apple M1, 20W). Geekbench 5. Compute Test. API OpenCL.



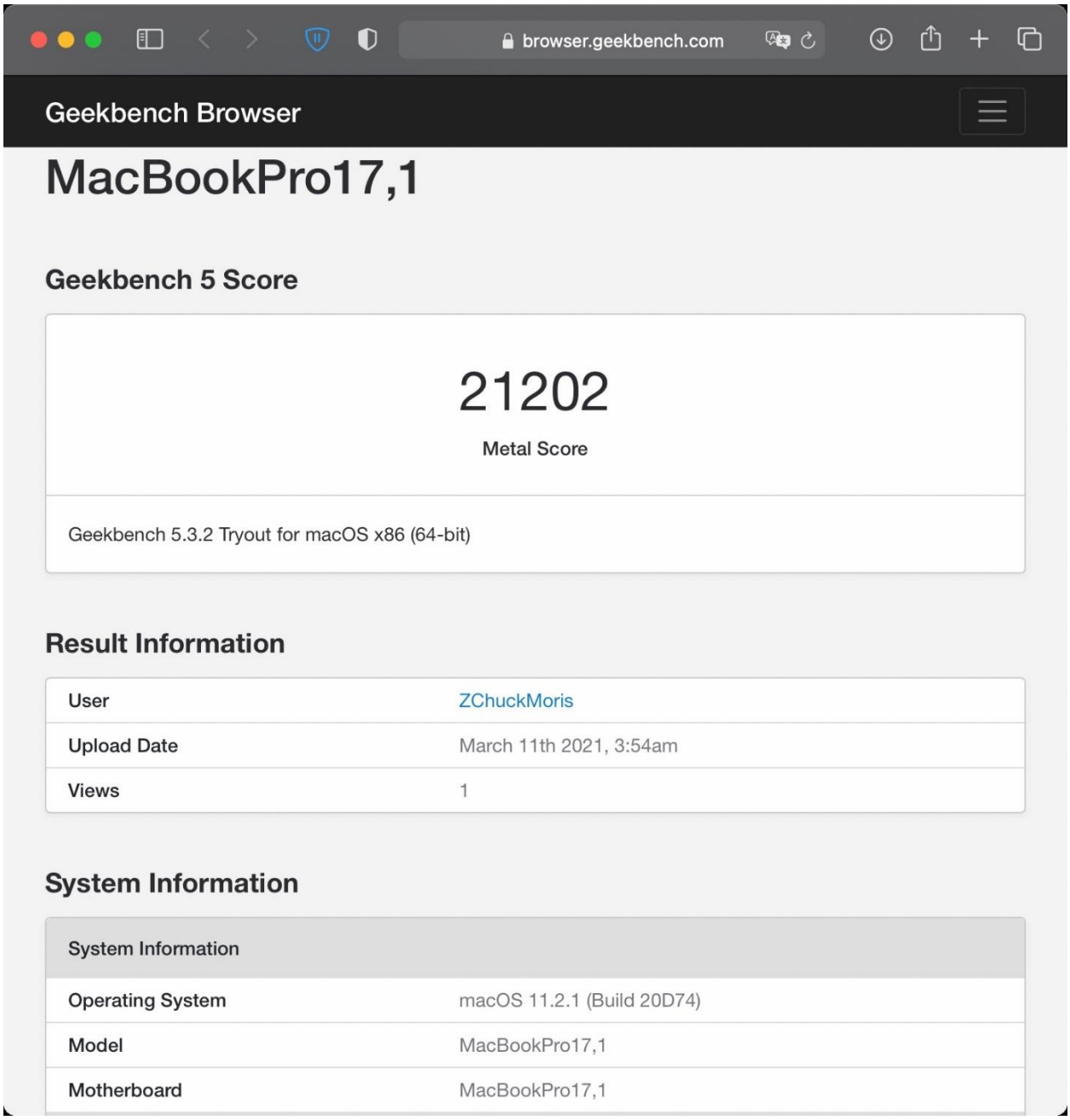
Скриншот 132. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Geekbench 5. Compute Test. API OpenCL.

С OpenCL картина примерно та же, если сравнивать с моделью 2020 года с 4 портами Thunderbolt 3 (разница в 2.2 раза вместо 2.08, но разница не критична), а вот если сравнивать с 8-ым поколением Intel, то разница уже сокращается и в случае с моделью 2019 года с 2 портами Thunderbolt 3 (i5 8257U) составляет не 3 раза, а 2.5 раза. Аналогично и с моделью 2018 года с 4 портами Thunderbolt 3 (i5 8259U). А если сравнить с моим Xiaomi, разница составляет 3.7 раза.

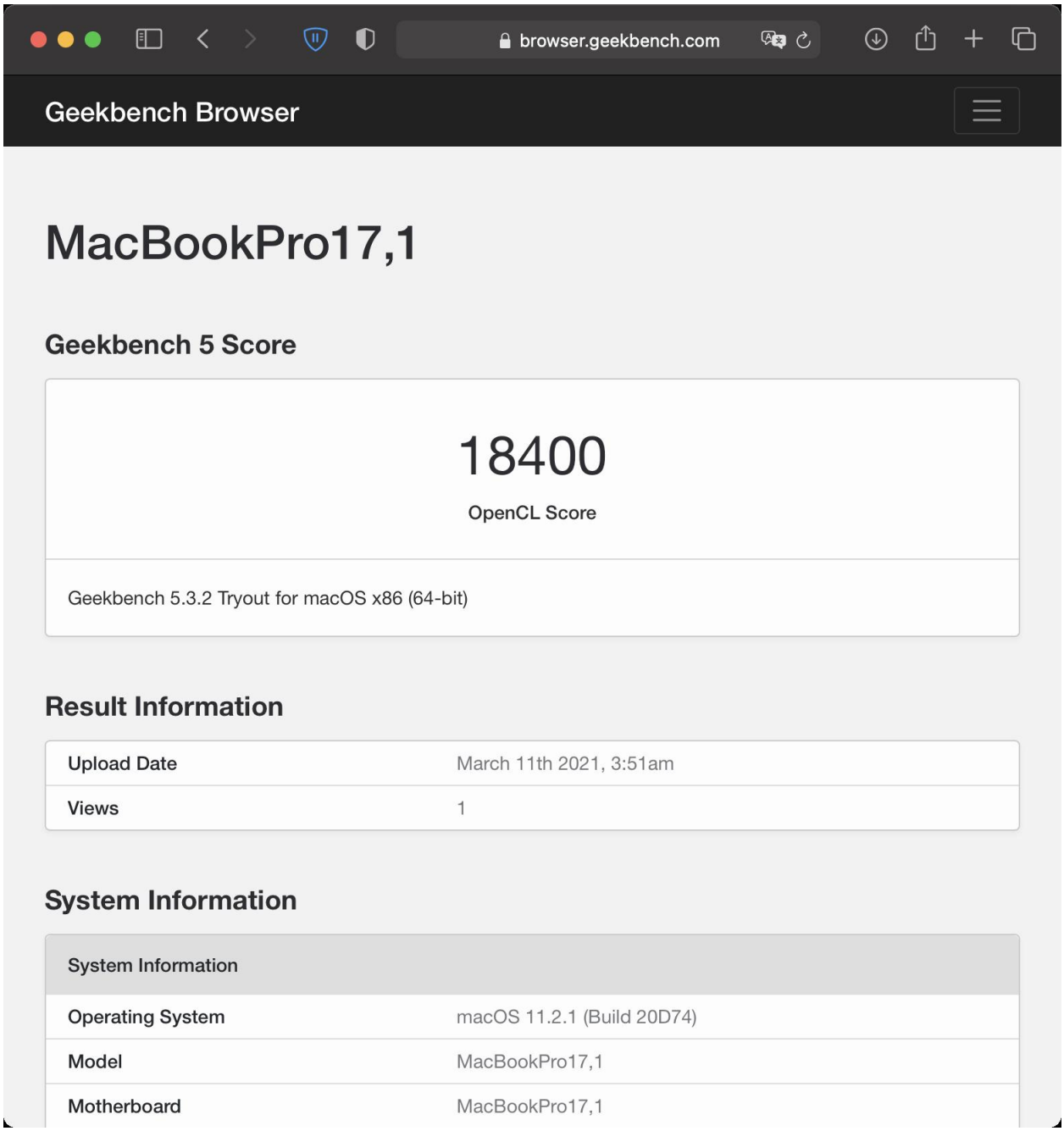
У Macbook Pro 13 2018 с 4 портами Thunderbolt 3 (8259U), как и у Macbook Pro 13 2019 с 2 портами Thunderbolt 3 (8257U), если верить тестам в Geekbench 5, в 1.5 раза более производительный графический ускоритель (Iris Plus 655 или 645) в сравнении с интегрированным в SoC от Intel в Xiaomi (UHD 620). Впрочем, и UHD 620 потребляет меньше энергии, всего 15 Ватт.

Я поясню, почему результаты Macbook Pro 13 с Core i5 взяты из сети: мои хорошие друзья не хотят сжечь свои ноутбуки ради тестов, результаты которых и так есть в интернете. Поэтому оппонентом Mac на M1 местами будет выступать Xiaomi. У него охлаждение чуть лучше, чем у Mac с 4x TB3.

Ещё 1 хороший вопрос: а насколько Rosetta 2 влияет на производительность видеоускорителя? Да, она транслирует команды из x86 в ARM, и делает это не очень эффективно, т.к., очевидно, на сегодняшний день невозможно практически без потерь в эффективности исполняемого машинного кода транслировать его с одной архитектуры в другую. Но, всё-таки, на графический производительности то Rosetta как сказывается?

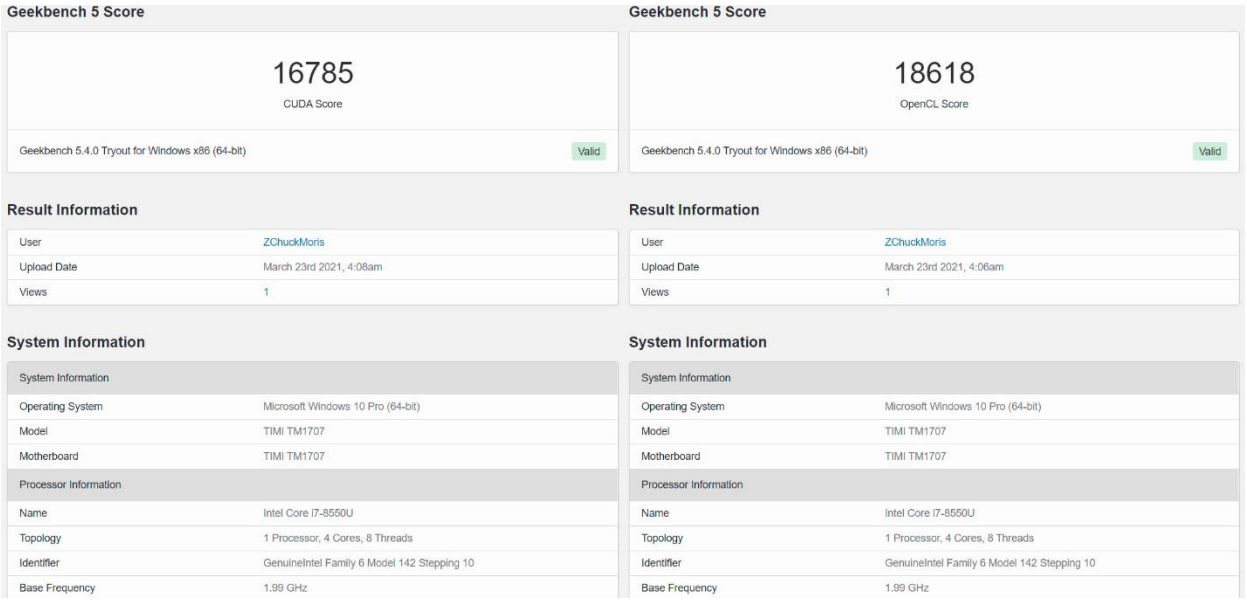


Скриншот 133. Macbook Pro 13 2020 (Apple M1, 20W). Geekbench 5. Compute Test. API Metal. Rosetta 2.



Скриншот 134. Macbook Pro 13 2020 (Apple M1, 20W). Geekbench 5. Compute Test. API OpenCL. Rosetta 2.

Если разница в 2.4% в случае с API Metal и 5% в случае с OpenCL для вас критична, то да, разница есть, и она существенная. А если спросить меня, то вообще плевать. Это было вполне ожидаемо, ведь от процессора то только и требуется, что направить данные на обработку видеокарте. Разница в 30% по части эффективности исполняемого на нём машинного кода не особо влияет на производительность графического ускорителя. Хотя, возможно, вы ощутите это в задачах, где требуется быстрый обмен данными между процессором и видеоускорителем. Проверим это ещё в нескольких сценариях.



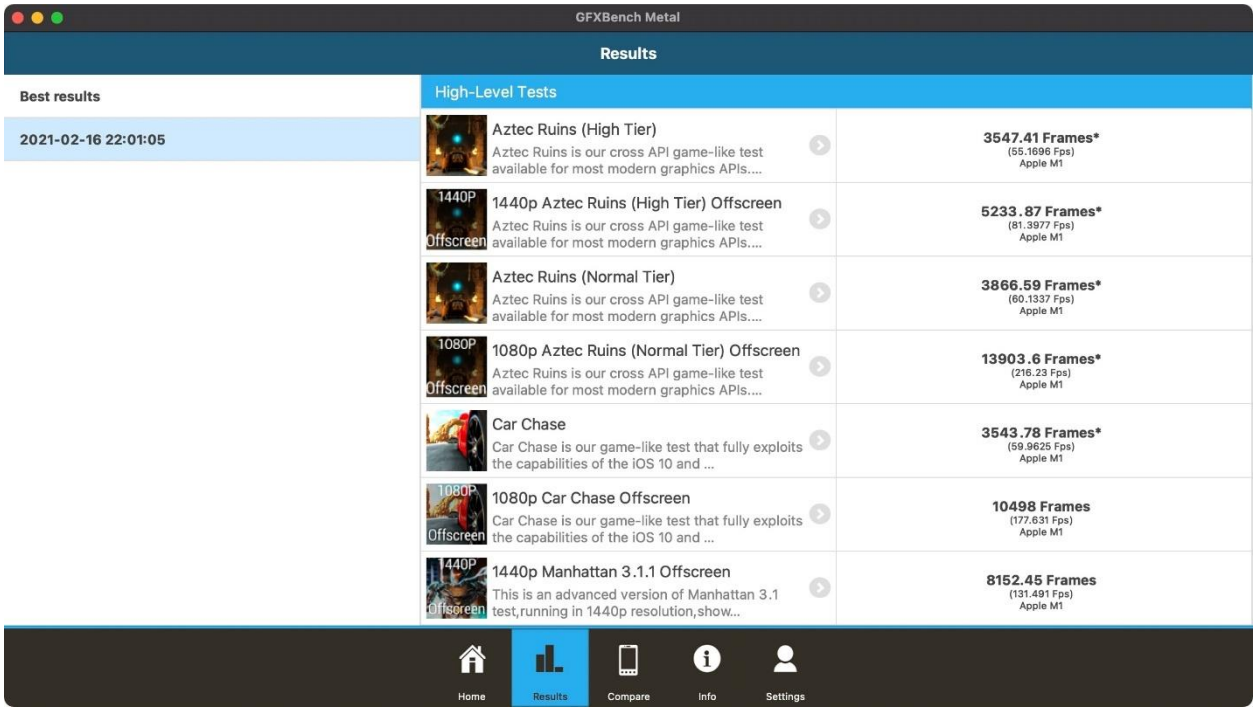
Скриншот 135. Результаты Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W) в Geekbench 5 Compute. ОС Windows 10 1909.

Остаётся ещё 1 открытый вопрос: а насколько производителен GPU в Apple M1 в сравнении с GTX 1050 Max-Q в моём Xiaomi Mi Notebook Pro GTX? Они практически одинаковые по части производительности, если сравнивать их в задаче Compute в Geekbench при помощи API OpenCL, разница между ними не превышает 4%. Я не знаю, можно ли напрямую сравнивать между собой Apple M1 с API Metal и GTX 1050 Max-Q с API CUDA. Много вопросов: а насколько корректно бенчмарк в Geekbench 5 задействует аппаратные возможности моей видеокарты GTX 1050 Max-Q при использовании API CUDA, нет ли дополнительных оптимизаций у этого теста специально под API Metal, и т.д. Но в целом выходит так, что GPU на 8 графических ядер в Apple M1 по своей производительности не уступает GTX 1050 Max-Q в моём Xiaomi, который потребляет при этом 30 Ватт энергии, а не 10. Феноменальный результат для интегрированной графики. Apple, вы меня удивляете.

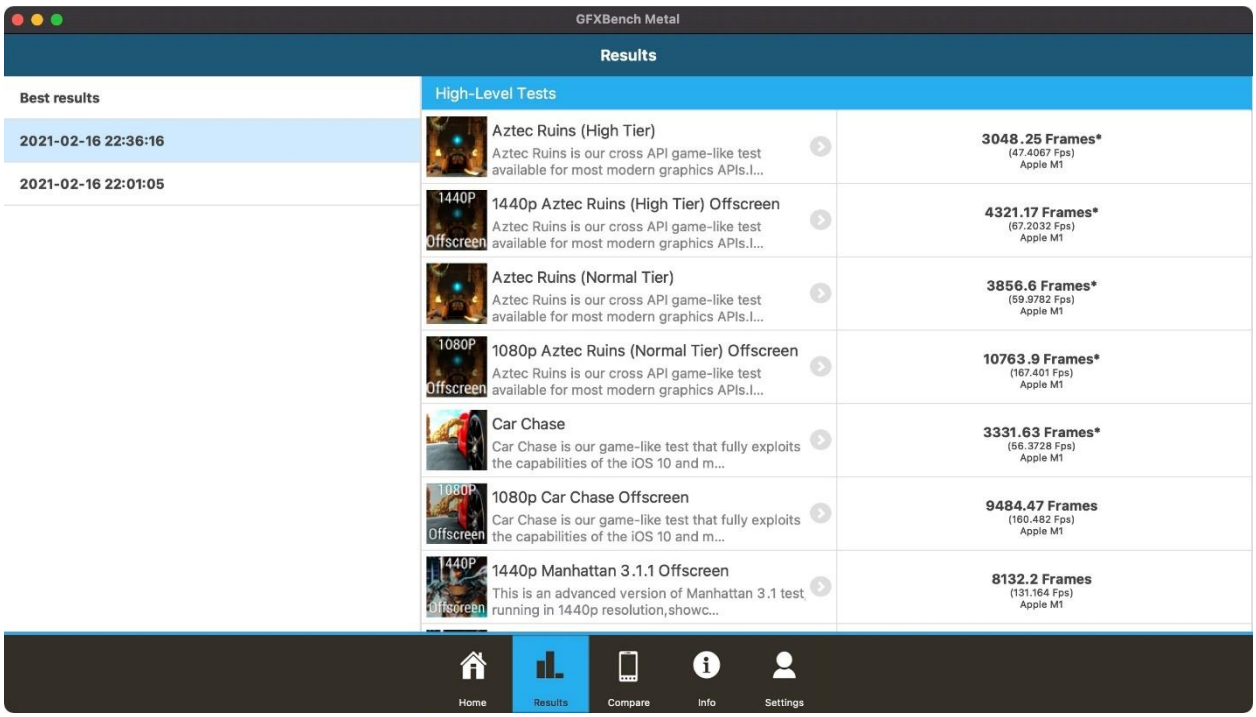
Дискретная видеокарта в моём ноутбуке Xiaomi не может быть задействована в хакинтоше, как и в большинстве ноутбуков с технологией NVidia Optimus для переключения между интегрированной и дискретной видеокартой. Из-за этого тесты с GTX 1050 Max-Q я провожу в Windows 10 на своём ноутбуке Xiaomi, иначе тут никак. Я понимаю, что не совсем корректно сравнивать разные железо на разных ОС, но тут уж ничего не поделать. Ноутбуков Apple с сопоставимым GPU у меня под рукой нет.

6.2. GFXBench (API Metal).

В GFXBench мы также взглянем на то, есть ли разница с Rosetta 2 и без.



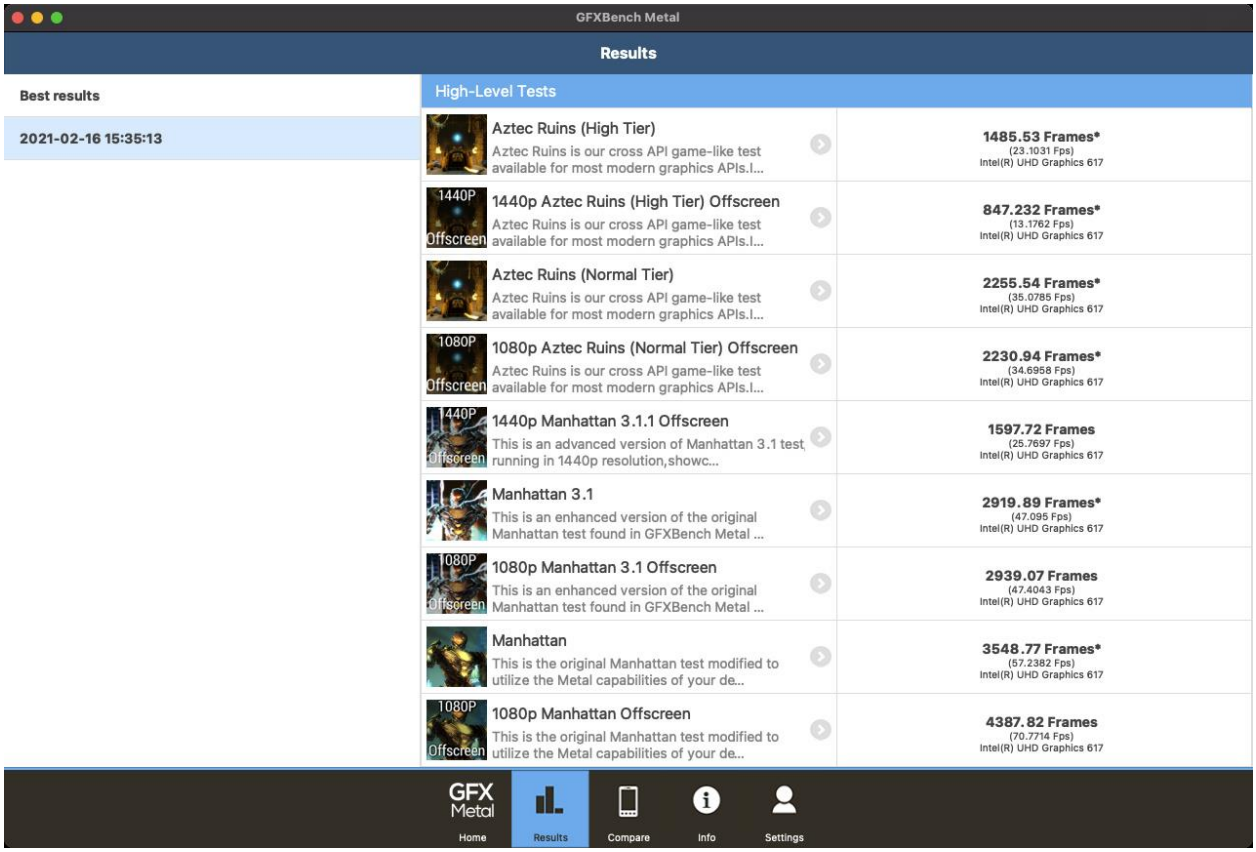
Скриншот 136. Macbook Pro 13 2020 (Apple M1 GPU, 10W). GFXBench. API Metal.



Скриншот 137. Macbook Pro 13 2020 (Apple M1 GPU, 10W). GFXBench. API Metal. Rosetta 2.

С Rosetta 2 в случае с GFXBench и сценой Aztec Ruins в режиме Offscreeen в 1440p разница составляет 21.1%, а в 1080p – 29.2%. В принципе тут довольно закономерно: чем выше разрешение, тем реже графический ускоритель отрисовывает кадры, и тем реже процессору нужно передавать видеоадаптеру данные для отрисовки этих самых кадров. Соответственно и

наоборот, чем разрешение ниже, тем ниже нагрузка на видеоускоритель и тем чаще она способна отрисовывать кадры, а информацию для отрисовки этих кадров передаёт процессор. Чем больше кадров в секунду надо отрисовать видеоадаптеру, тем больше нагрузка на ЦП, и так пока нагрузка на ЦП или ГП не достигнет 100%. И да, в случае со сценой Aztec Ruins в GFXBench нужен быстрый и постоянный обмен данными между видеоускорителем и процессорными ядрами, оттого и разница между Rosetta 2 и нативным запуском достигает аж почти 30% в 1080p. То же самое касается Driver overhead 1080p Offscreen. В этой задаче нужно быстро передавать данные видеоускорителю, и там разница становится уже почти двукратной.



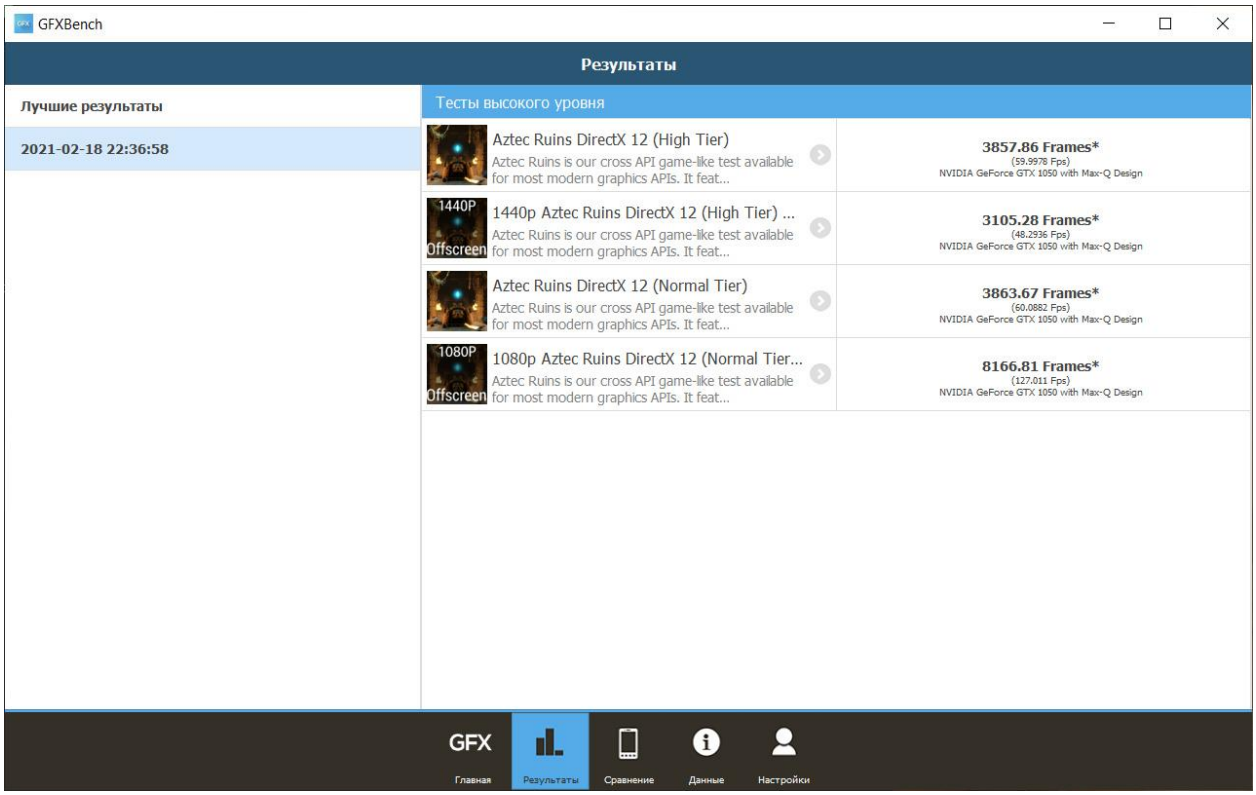
Скриншот 138. Xiaomi Mi Notebook Pro GTX (Intel UHD 620, 15W). GFXBench. macOS. API Metal.

Разница между M1 и интегрированной графикой в Xiaomi – просто небо и земля. В Offscreen 1440p и 1080p сцене Aztec Ruins разница в пользу графики в M1 составила 517-523%, т.е. более 6 раз! Это феноменальный отрыв. Может быть, у меня Xiaomi просто бракованный?

А, нет, находим в базе GFXBench результаты теста графики с API Metal для Intel Iris Plus Graphics 655 (Macbook Pro 13 2018 с 4x Thunderbolt 3) и Intel

Iris Plus Graphics 645 (Macbook Pro 13 2019 с 2x Thunderbolt3), видим, что в сцене Aztec Ruins отрыв M1 от 1-го составил 267% для 1080p и 269% для 1440p, а от 2-го (с 2 портами Thunderbolt 3) 294% и 291% соответственно.

Более того, обратите внимание на результат ноутбука Xiaomi ОС Windows при использовании DirectX 12.



Скриншот 139. Xiaomi Mi Notebook Pro GTX (GTX 1050 Max-Q, 30W). GFXBench. Windows. API DirectX 12.

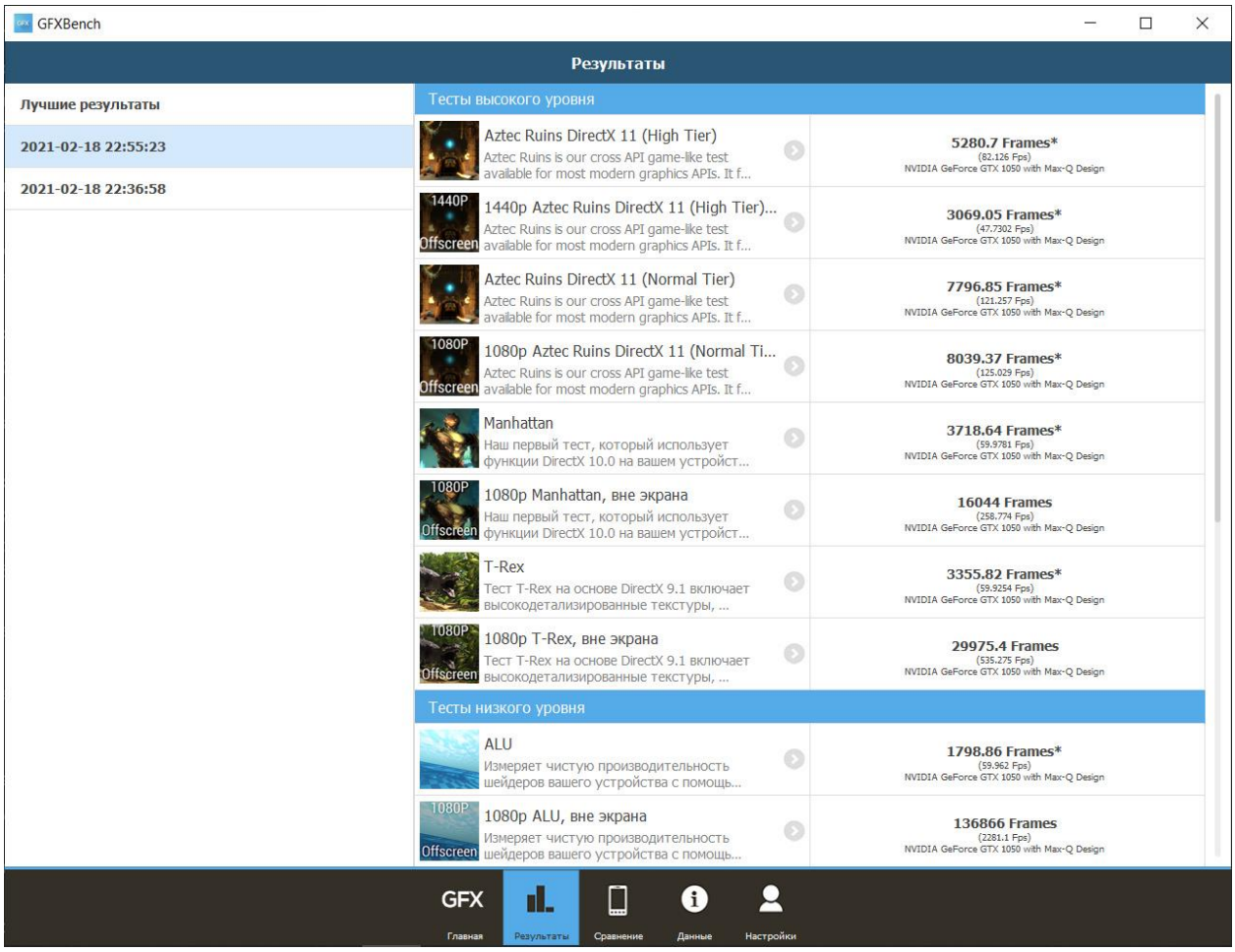
Напоминаю, что в Windows, в отличие от macOS, на моём Xiaomi функционирует дискретная видеокарта GTX 1050 Max-Q с TDP 30 Ватт. На macOS она работать не может, т.к. macOS не поддерживает технологию NVidia Optimus, на базе которой в большинстве современных Windows ноутбуков и реализована связка интегрированной графики + дискретной графики NVidia. Apple же использует свою собственную реализацию, и эта реализация прописана на уровне BIOS/UEFI, и как-либо её сымитировать у хакинтошеров пока не выходит.

И ещё одно важное уточнение: это разные операционные системы и разные графические API, и вряд ли ещё одинаково эффективно GFXBench задействует разные API, так что лоб в лоб по идее их сравнивать нельзя.

Но как-либо ещё сопоставить эти видеоускорители будет сложно, потому прибегаем к такому сопоставлению. В сцене Aztec Ruins

видеоускоритель в M1, который потребляет всего-то жалкие 10 Ватт энергии и выделяет соответствующее количество тепла, обошёл GTX 1050 Max-Q на 68,5% в 1440p и 70% в 1080p.

Apple, как?! Как Вы вместили мощь дискретной видеокарты начального уровня в интегрированный ускоритель с TDP 10 Ватт? Объясните мне. И ещё больше я недоумеваю от того, почему же у чипа M1, который прекрасен и является, с моей сугубо личной точки зрения, произведением гениальной инженерной мысли, оснащён настолько ужасным охлаждением?



Скриншот 140. Xiaomi Mi Notebook Pro GTX (GTX 1050 Max-Q, 30W). GFXBench. Windows. API DirectX 11.

И с использованием DirectX 11, и с использованием OpenGL, GTX 1050 Max-Q в ноутбуке Xiaomi Mi Notebook Pro GTX никак не может ни в одной высокоуровневой сцене тягаться с интегрированной графикой в Apple M1.

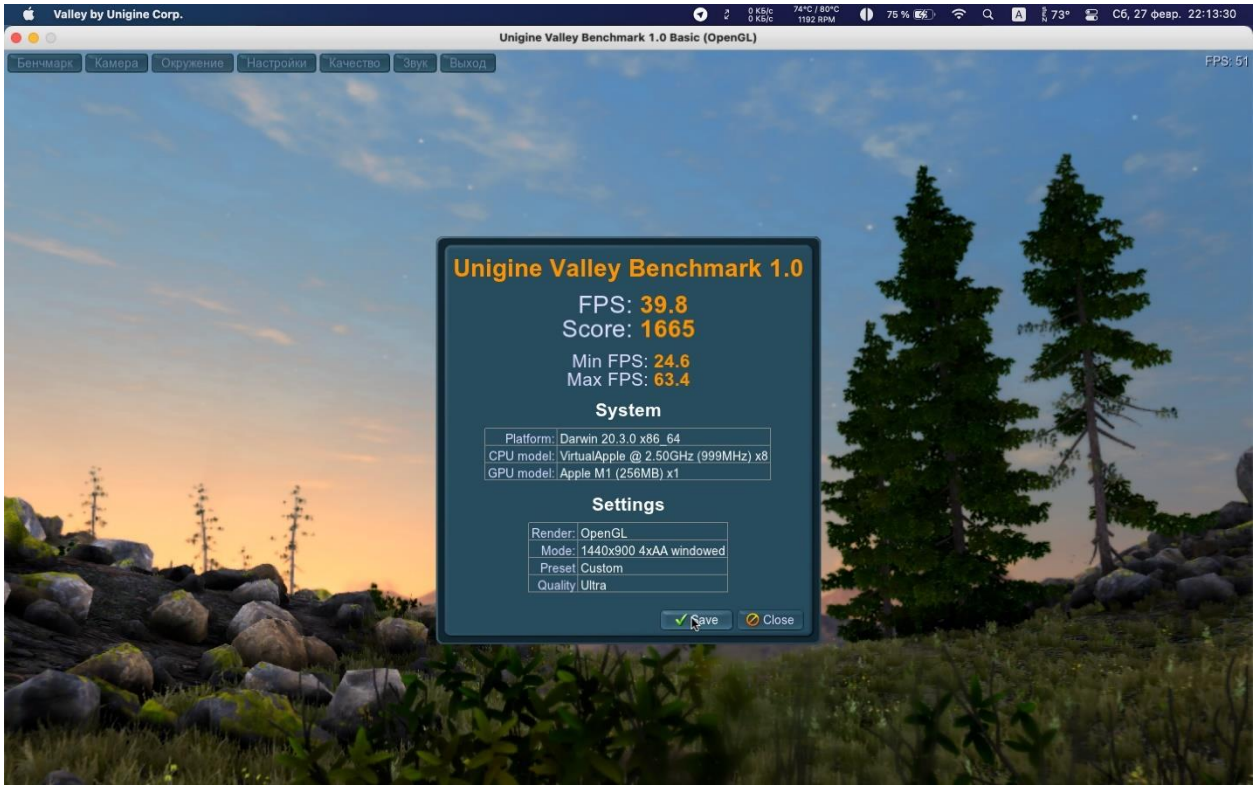
Разве что в Onscreen может где-то проступать паритет, и то только за счёт того, что в Onscreen имеет место вертикальная синхронизация, и меньшее разрешение экрана у Xiaomi даёт ему фору в этом сценарии (экран то у него 1920x1080, а не 2560x1600, как у Macbook Pro 13”).

6.3. Unigine Valley (API OpenGL).

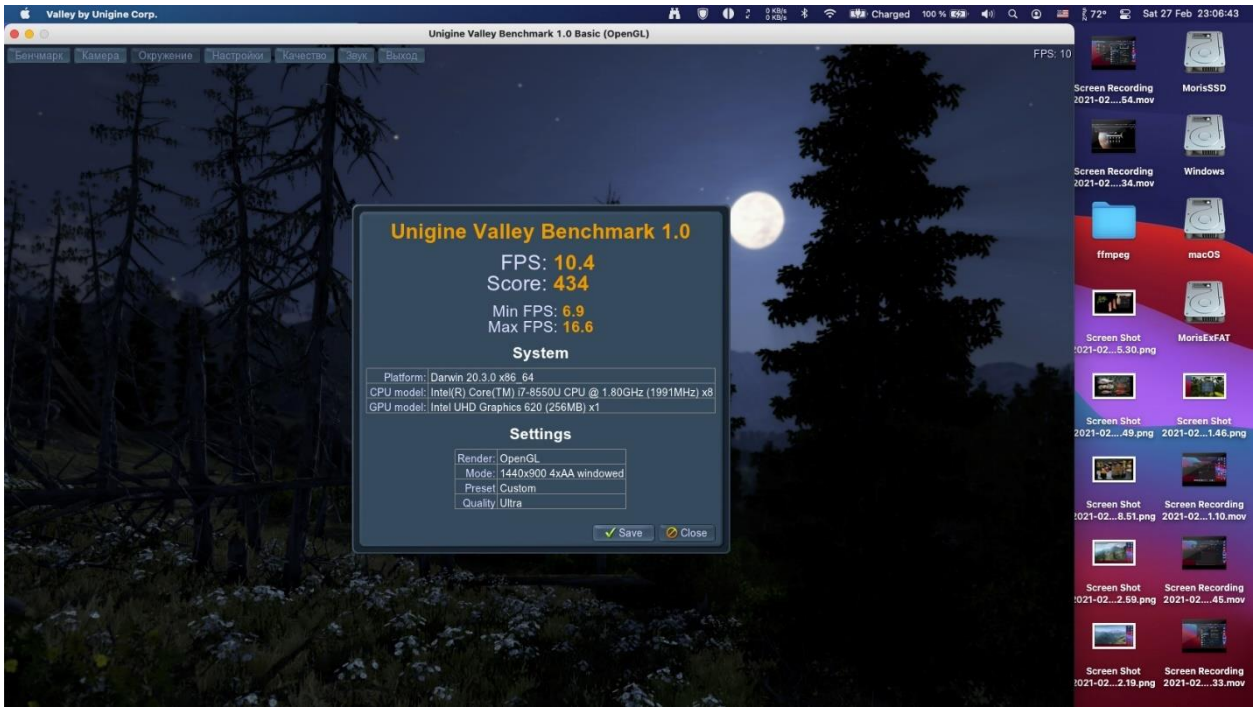
Вот результатов этого теста в сети Вы не найдёте. Я выше в главе 2.4 объяснял, почему с приложениями от Unigine есть проблемы: они мало того, что работают через Rosetta 2, что усложняет процессору (в сравнении с нативно скомпилированным под него кодом) задачу передачи информации для отрисовки кадров видеокарте, так там ещё и используется API OpenGL, от поддержки которого Apple отказались ещё 2 года назад, и с которым всплыло значительное число проблем в macOS Big Sur и особенно с новым SoC Apple M1.

Тестировать мы будем в окне, т.к. разрешение экрана и соотношение сторон у моего Xiaomi отличаются от таковых в Macbook. Тестируем с разрешением 1440x900 (с масштабированием при этом разрешении окно занимает практически весь экран на Macbook Pro 13 с чипом Apple M1), настройками качества Ultra и сглаживанием 4xAA (со сглаживанием 8xAA Unigine Valley не работает на Macbook Pro с SoC Apple M1 из-за проблем с работой OpenGL на новой macOS Big Sur с новым SoC Apple M1).

И если даже здесь Macbook Pro 2020 на Apple M1 вырвется вперёд, это финита ля комедия для старых SoC Intel. И спойлер: да, он вырвался вперёд.



Скриншот 141. Macbook Pro 13 2020 (Apple M1 GPU, 10W). Unigine Valley Benchmark. 900p, Ultra качество, 4xAA.



Скриншот 142. Xiaomi Mi Notebook Pro GTX (UHD 620, 15W). Unigine Valley Benchmark. 900p, Ultra качество, 4xAA.



Скриншот 143. Macbook Pro 13 2020 4x TB3 (Iris Plus G7). Unigine Valley Benchmark. 900p, Ultra качество,4xAA.

Я попросил Владимира Баруса протестировать свой Macbook Pro 13 2020 с 4 портами Thunderbolt 3 (i5 1038NG7) в этом бенчмарке. Как Вы видите, интегрированная графика в новом Macbook Pro 13 с чипом M1 в 2,07 раза превосходит Iris Plus G7 в Macbook Pro Владимира и в 3,84 раза превосходит Intel UHD Graphics 620 в моём ноутбуке Xiaomi (i7 8550U).

Разница просто ошеломительная даже в худшем сценарии для M1. Да, она не столь велика в сравнении с интегрированной графикой Xiaomi, сколь она была велика в предыдущих тестах, где задействовался API Metal, но она в любом случае невероятно высока.

А если запустить это тест на Windows и задействовать видеокарту GeForce GTX 1050 Max-Q в ноутбуке Xiaomi?

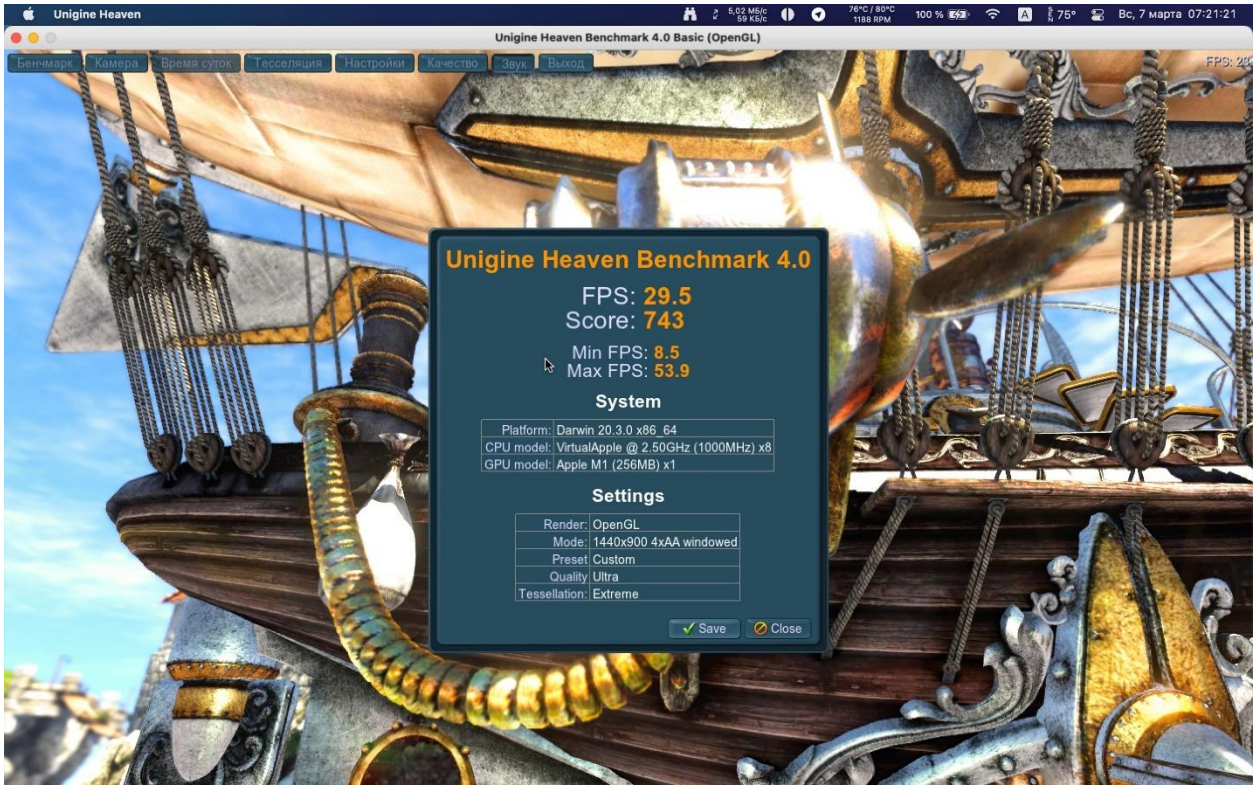


Скриншот 144. Xiaomi Mi Notebook Pro GTX (GTX 1050 Max-Q). Unigine Valley Benchmark. 900p, Ultra качество, 4xAA.

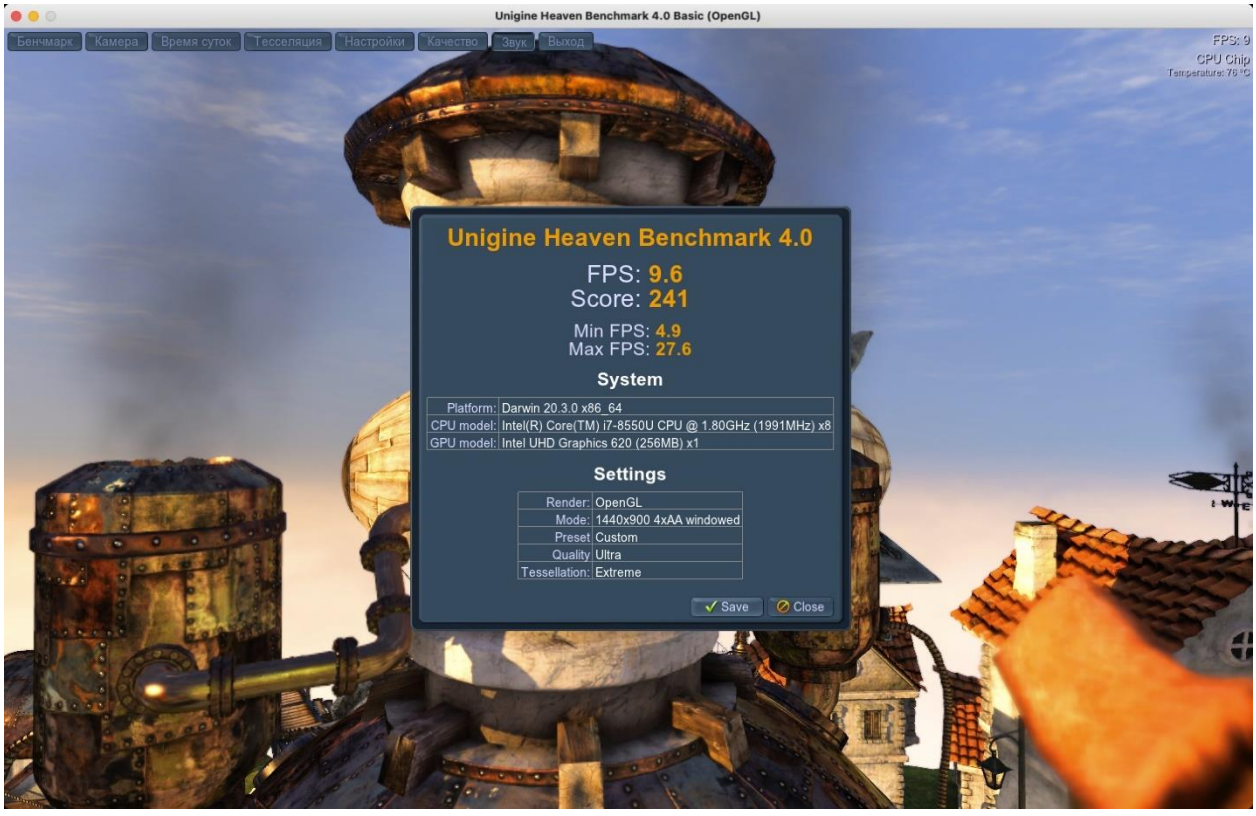
На Windows при использовании дискретного видеоускорителя NVidia GeForce GTX 1050 Max-Q мой ноутбук Xiaomi в бенчмарке Unigine Valley отрисовал на 27,5% больше кадров при использовании API OpenGL. С DirectX11 там разница мала, всё в пределах погрешности. Вывод по Unigine Valley: графический ускоритель в M1 по производительности сопоставим с GTX 1050 Max-Q. Вы можете задаться вопросом: почему при разнице аж в 27.5% между Apple M1 и GTX 1050 Max-Q, я считаю их сопоставимыми? Ответ прост: OpenGL и Rosetta 2. Мы с Вами выше на примере с GFXBench убедились, что более высокие задержки при обмене данными между центральными ядрами процессора и графическим ускорителем, вызванные не столь эффективным транслированным кодом, могут снизить FPS на ± 30%.

Учтите ещё, что Unigine Valley задействует OpenGL, работу которого на macOS 10.14+ никто не гарантирует, также и в случае с Unigine Heaven.

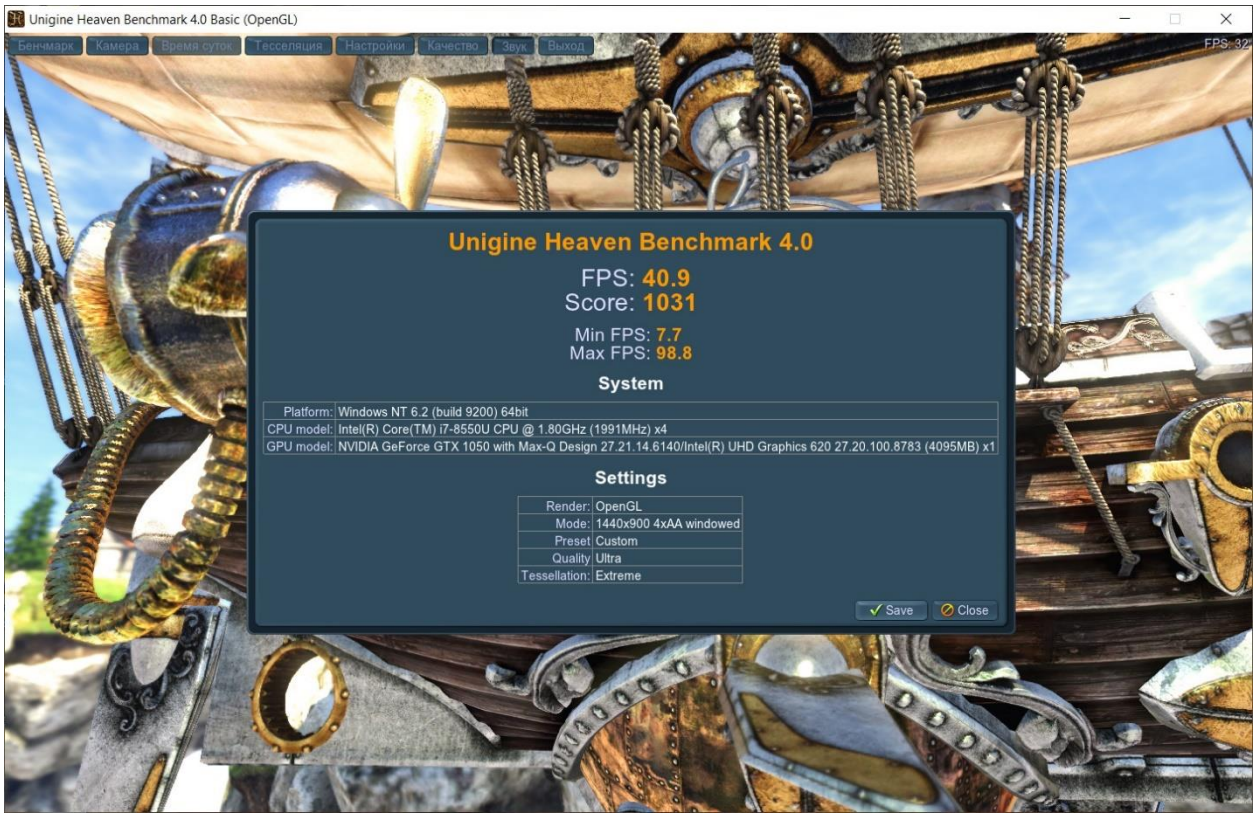
6.4. Unigine Heaven (API OpenGL).



Скриншот 145. Macbook Pro 13 2020 (Apple M1 GPU, 10W). Unigine Heaven Benchmark. 900p, Ultra качество, 4xAA.



Скриншот 146. Xiaomi Mi Notebook Pro GTX (UHD 620, 15W). Unigine Valley Benchmark. 900p, Ultra качество, 4xAA.



Скриншот 147. Xiaomi Mi Notebook Pro GTX (GTX 1050 Max-Q). Unigine Heaven Benchmark. 900p, Ultra качество, 4xAA.

Тут разница ещё больше между GTX 1050 Max-Q и GPU в Apple M1: 38.8%, если за 100% принимать показатели Apple M1. Разница же между GPU в Apple M1 и GPU в i7 8550U (UHD 620) составила 3,08 раза или 208%.

И в этом нет ничего особо удивительного. Чем больше FPS, тем больше и разница, т.к. чаще происходит обмен данными между центральными ядрами процессора и графическим ускорителем. А код, который исполняется на процессоре, не столь эффективен, т.к. он не изначально скомпилирован под родную архитектуру нового Mac с SoC Apple M1, aarch64 (ARM64), а транслирован из x86_64. Предварительная трансляция – дело хорошее, но транслированный по итогу код в любом случае не сравнится по части эффективности исполнения с нативно скомпилированным кодом. Вот поэтому Вы и наблюдаете такую разницу.

И, разумеется, не забываем, что здесь используется OpenGL, с которым у M1 всё не очень хорошо и, скорее всего, также будет не очень хорошо в будущем и у старых Mac с SoC Intel и GPU Intel (iGPU) или AMD (dGPU) с последующими обновлениями ОС. Гарантий по части OpenGL никаких нет и не будет. Вы или соглашаетесь с этим, или не покупаете Mac. Всё просто.

6.5. Сравнительная таблица с результатами тестирования GPU.

Specs	Laptop/PC	Xiaomi Mi Notebook Pro GTX GTX 1050 Max-Q (30W) CUDA	Xiaomi Mi Notebook Pro GTX Intel UHD 620 (15W) macOS	HP Omen 15 15-en0037ur GTX 1660 Ti (80W) CUDA	Lenovo Yoga Slim 7 Radeon RX Vega 8 Extreme Performance 15W	Apple Macbook Pro 13 2020 Z11B0004T (Apple M1) Rosetta 2 10W	Apple Macbook Pro 13 2020 Z11B0004T (Apple M1) Native 10W
Geekbench 5	OpenCL	18618	5202	63149	14734	18400	19329
	Vulkan			52284	16663		
	CUDA	16785		66489			
	Metal		4947			21202	21709
GFXBench 5.0 Metal (frames / Mtexel/s / mB PSNR)	Aztec Ruins 1440p High Tier		1486			3048	3547
	Aztec Ruins 1440p High Tier Offscreen		847			4321	5234
	Aztec Ruins 1080p Normal Tier		2256			3857	3867
	Aztec Ruins 1080p Normal Tier Offscreen		2231			10764	13904
	Car Chase					3332	3544
	Car Chase 1080p Offscreen					9484	10498
	Manhattan 3.1.1 1440p Offscreen		1597			8132	8152
	Manhattan 3.1		2920			3722	3722
	Manhattan 3.1 1080p Offscreen		2939			15906	17006
	Manhattan		3549			3725	3724
	Manhattan 1080p Offscreen		4388			24674	25068
	T-Rex		3356			3364	3364
	T-Rex 1080p Offscreen		8114			28202	36270
	ALU 2		1799			1817	1817
	1080p ALU 2 Offscreen		7319			17956	17955
	Driver Overhead		1802			1799	1801
	Driver Overhead 1080p Offscreen		11055			7503	14759
	Texturing		10822			70914	70929
	Texturing 1080p Offscreen		11607			71186	71222

Таблица 3. Сравнительная таблица с результатами тестов GPU.

Выводы по производительности GPU: среди интегрированных решений GPU тут быстрейший, но он не сравнится с дискретной видеокартой уровня GTX 1660 Ti Mobile или RTX 2060 Max-Q. Но при энергопотреблении в 10 Ватт GPU в M1 стоит на уровне GTX 1050 Max-Q. Это, однозначно, успех. Хотя, вполне возможно, что AMD при 5 нм тоже так смогли бы.

7. Тестирование блоков аппаратного кодирования и декодирования видео.

7.1. А что заявлено по аппаратным блокам у Apple M1?

Информации за время тестирования я смог собрать немало и хотел бы ей с Вами поделиться. Сразу скажу: в Final Cut я не работаю, как и в Davinci Resolve, и в Adobe Premiere Pro. Для проведения этого теста я обычно использую консольную утилиту ffmpeg, которая в качестве библиотеки входит в состав множества приложений с открытым исходным кодом, включая HandBrake, MP4Tools, OBS и многих других.

Т.к. многие приложения полагаются на ffmpeg при кодировании и декодировании видео, можно предположить, что если в ffmpeg в чистом виде наблюдается прирост в производительности, то он будет и в приложениях, которые задействуют ffmpeg в работе. Тут всё просто.

Но прежде, чем мы перейдём к тестированию, я бы хотел разобрать один спорный момент в видео Романа Ястреба «🏆Революция ARM - Узнаешь все про Apple Silicon!🤓», которое вышло 3 апреля. На 7:56 Роман сообщает, что SoC Apple M1 имеет аппаратное декодирование видео в 8K.

Напоминаю, что я со всем уважением отношусь к Роману и я ни в коем ради не пытаюсь как-либо его обидеть или самоутвердиться за счёт него. Мне понравился его ролик, его ролики про компьютеры Mac одни из самых подробных в российском сегменте YouTube, так что не мудрено, что среди множества деталей находятся ошибки, которые я и хочу сейчас разобрать.

Что же здесь спорного? Да, в общем-то, то, что сами Apple этого нигде не указывают. Я нашёл лишь информацию на сайте Apple для разработчиков, где указаны лишь опции, которые разработчики могут использовать для задействования аппаратного или программного декодирования видео для разных кодеков: тут и H.264, и H.265, и VP9, и ProRes и т.д..

Проблема в том, что там не указано разрешение, при котором аппаратные блоки декодирования видео могут обработать это самое видео. Такой информации Apple нигде не указывает.

NVENC Summary[6][9]

GPU Hardware			H.264 (AVC) (In H.264, NVENC always has B Frame support, max 4096x4096 resolution, and max 8-bit depth)			H.265 (HEVC)					
NVENC Generation	GPU Code Name	NVENC per Chip	Chroma		Lossless Coding	Chroma		Lossless Coding	Resolution	Color Depth	B Frames
			4:2:0	4:4:4		4:2:0	4:4:4				
1st Gen	GK110	1	✓	✗	✗	H.265 not supported					
	GK107										
	GK106										
	GK104										
2nd Gen	GM108	0	No NVENC encoders available								
3rd Gen	GM107	1	✓	✓	✓	H.265 not supported					
	GM208	1	✓	✗	✗						
	GM206			✓	✓	✓	Y?[9] N?[8]	Y?[9] N?[8]	4096 x 4096	8-bit	✗
	GM204	2					✗	✗			
	GM200										
4th Gen	GP108	0	No NVENC encoders available								
	GP107	1	✓	✓	✓	✓	✓	✓	8192 x 8192	10-bit	✗
	GP106										
	GP104-2xx+	2									
	GP104-1xx	1									
	GP102	2									
	GP100	3							4096 x 4096		
5th Gen	GV10x										
6th Gen	TU117	1							8192 x 8192		✓
	TU116										
	TU106										
	TU104										
	TU102										
7th Gen	GA104	1	✓	✓	✓	✓	✓	✓	8192 x 8192	10-bit	✓
	GA102										
	A100[10]	0	No NVENC encoders available								

Скриншот 148. Информация о поддерживаемых кодеках для кодирования видео видеокартами NVidia.

NVidia же, напротив, указывает всю нужную информацию не только по части декодирования видео, но и по части кодирования видео, благодаря чему затем на сторонних ресурсах появляются сравнительные таблицы с возможностями аппаратного кодирования видео моделями видеокарт NVidia. Для понимания, GA104, который Вы можете видеть выше, это чип, который устанавливается в видеокарты GeForce 30-ой серии, в числе которых и ноутбучные RTX 3080 Max-Q, и RTX 3080 Mobile, и RTX 3070 Max-Q, и RTX 3070 Mobile, последний из которых мы тестировали совсем недавно в обзоре Asus ROG Strix G15.

NVDEC (Decoding) matrix for NVIDIA GPUs

	*H.265 (HEVC) 4:4:4		H.265 (HEVC) 4:2:0		H.264 (AVCHD) 4:2:0	
GPU	MAX Color	MAX Res.	MAX Color	MAX Res.	MAX Color	MAX Res.
Kepler	N/A	N/A	N/A	N/A	8-bit	4096 x 4096
Maxwell (1st Gen)*	N/A	N/A	N/A	N/A	8-bit	4096 x 4096
Maxwell (2nd Gen)	N/A	N/A	N/A	N/A	8-bit	4096 x 4096
Maxwell (GM206)	N/A	N/A	10-bit	4096 x 2304	8-bit	4096 x 4096
Pascal	N/A	N/A	12-bit	8192 x 8192**	8-bit	4096 x 4096
Volta	N/A	N/A	12-bit	8192 x 8192	8-bit	4096 x 4096
Turing	12-bit	8192 x 8192	12-bit	8192 x 8192	8-bit	4096 x 4096
Ampere (A100)	12-bit	8192 x 8192	12-bit	8192 x 8192	8-bit	4096 x 4096
Ampere (non A100)	12-bit	8192 x 8192	12-bit	8192 x 8192	8-bit	4096 x 4096

Table continued:

NVDEC (Decoding) matrix for NVIDIA GPUs

	VP9		VP8		MPEG-2		VC-1		AV1	
GPU	MAX Color	MAX Res.	MAX Color	MAX Res.	MAX Color	MAX Res.	MAX Color	MAX Res.	MAX Color	MAX Res.
Kepler	N/A	N/A	N/A	N/A	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Maxwell (1st Gen)*	N/A	N/A	N/A	N/A	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Maxwell (2nd Gen)	N/A	N/A	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Maxwell (GM206)	8-bit	4096 x 2304	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Pascal	12-bit****	8192 x 8192**	8-bit	4096 x 4096***	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Volta	12-bit	8192 x 8192	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Turing	12-bit	8192 x 8192	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Ampere (A100)	12-bit	8192 x 8192	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	N/A	N/A
Ampere (non A100)	12-bit	8192 x 8192	8-bit	4096 x 4096	8-bit	4080 x 4080	8-bit	2048 x 1024	10-bit	8192 x 8192

Скриншот 149. Информация о поддерживаемых кодеках для декодирования видео видеокартами NVidia.

Разумеется помимо аппаратного кодирования видео в 8K в кодеке H.265 (HEVC), также я нашёл на сайте videocardz таблицу с возможностями декодирования процессора, основанную на информации самой NVidia (NVidia Video Codec SDK и NVidia Decode/Encode Matrix). У видеокарт NVidia указывается информация о том, с какой глубиной цвета (8/10/12 бит) поддерживается кодирование/декодирование видео, указывается цветовая субдискретизация (4:4:4 у H.265 с RTX 3xxx), указывается максимальное разрешение видео (8192x8192 для H.265/HEVC, VP9 и AV1, 4096x4096 для H.264 и VP8, и т.д.), но в случае с Apple таких сведений нигде нет.

У AMD в случае с их Video Core Next информации поменьше, на сайте X.Org (разработчики наиболее популярного дисплейного менеджера для Linux) указана только общее по всем кодекам максимальное разрешение.

У Intel с их Quick Sync Video информации значительно больше, чем у AMD. В сравнительной таблице то информации примерно столько же, но при этом присутствует информация о внедрении блоков аппаратного декодирования AV1 8K 10-bit 4:2:0 видео в процессоры 11-го поколения Tiger Lake (10 нм) в модели с графикой 12-го поколения (например, Intel Iris Xe G7 на 80 или 96 вычислительных блоков) со ссылкой на релиз драйвера графики Intel для Linux в GitHub с соответствующей информацией. Также указана в таблице информация о поддержке SoC аппаратного декодирования видео H.265/HEVC в разрешении 8K.

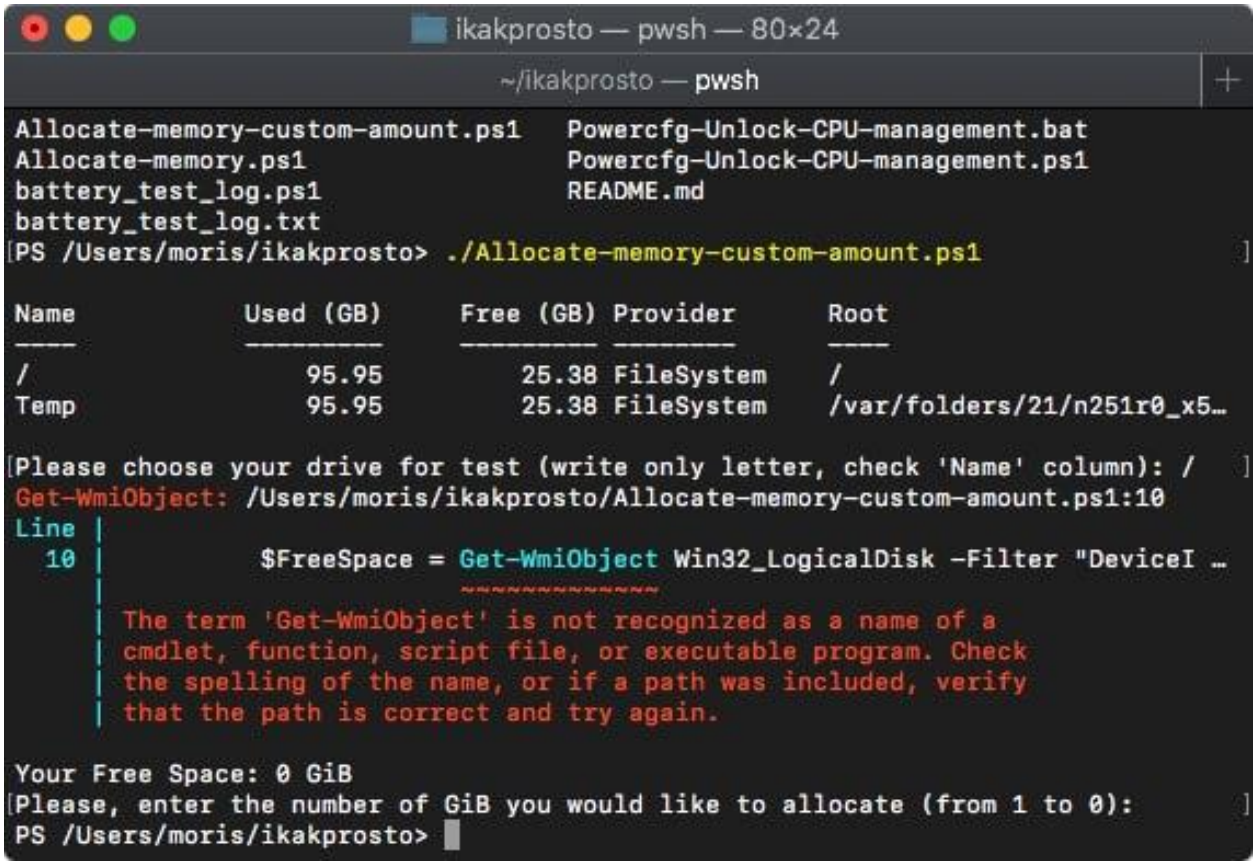
К чему я это всё? Да, у Apple M1 имеются аппаратные декодеры видео, но никто не заявлял, что они занимаются декодированием и 8K видео, нигде нет информации о поддерживаемом разрешении в каждом кодеке. Поэтому остаётся нам всё проверять лишь тестами. Но тут мы упираемся в вопрос совместимости с ПО: а оно умеет задействовать аппаратное ускорение для 8K видео через API Apple VideoToolBox? А само API Apple VideoToolBox позволяет уже работать с 8K? В macOS вообще поддержка 8K видео на аппаратном уровне была когда-либо реализована?

Одно дело – заявить, что ноутбук тянет редактирование ролика в 8K, а другое – что он его тянет благодаря аппаратным блокам декодирования видео. Это несколько разные понятия и тут была, скорее, даже оговорка у Романа. Хотя, вполне возможно, что они и умеют в 8K, но это ещё нужно проверить. Я постараюсь определить, каковы возможности Macbook Pro на Apple M1 и по части аппаратных блоков декодирования видео, но никаких гарантий не могу дать касательно того, что проблема явно не будет в каком-либо ПО, в котором мы будем тестировать это дело, или же в ОС, или в каком-то ещё третьем факторе. Ну, тут уж куда деваться, придётся отталкиваться от того, что есть, т.к. тема в самом деле интересная.

Сперва проведём тесты в ffmpeg, dav1d (что это за чудо, скоро узнаете), а затем уже в браузерах и VLC.

7.2. Тесты с кодеками H.265 и H.264 при помощи ffmpeg.

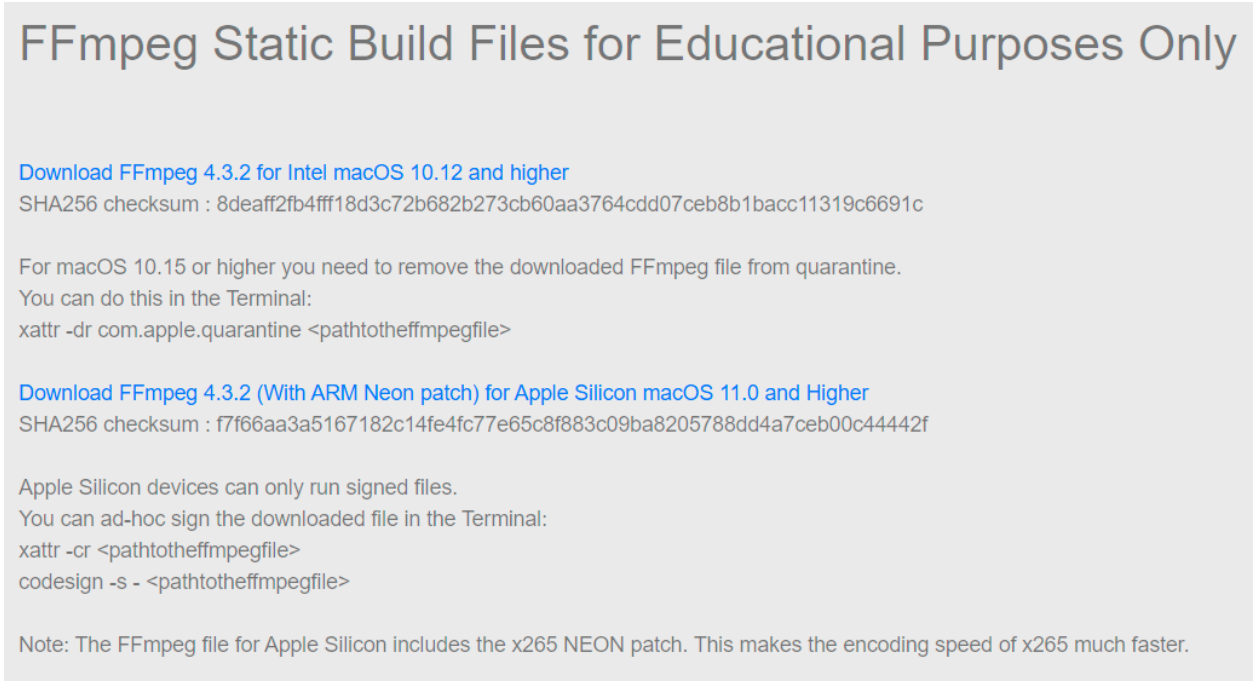
Мы попробуем преобразовать 4K 60 FPS H.265 8-bit 4:2:0 видео с битрейтом 77,4 Мбит/сек, и также проверим скорость его перекодирования из H.265 в H.264. Длительность видео – 02:59, размер – 1,61 GiB. Начнём.



Скриншот 150. PowerShell. Некорректная работа в macOS. Скриншот с Xiaomi, macOS 10.15.7 запущена с другого SSD.

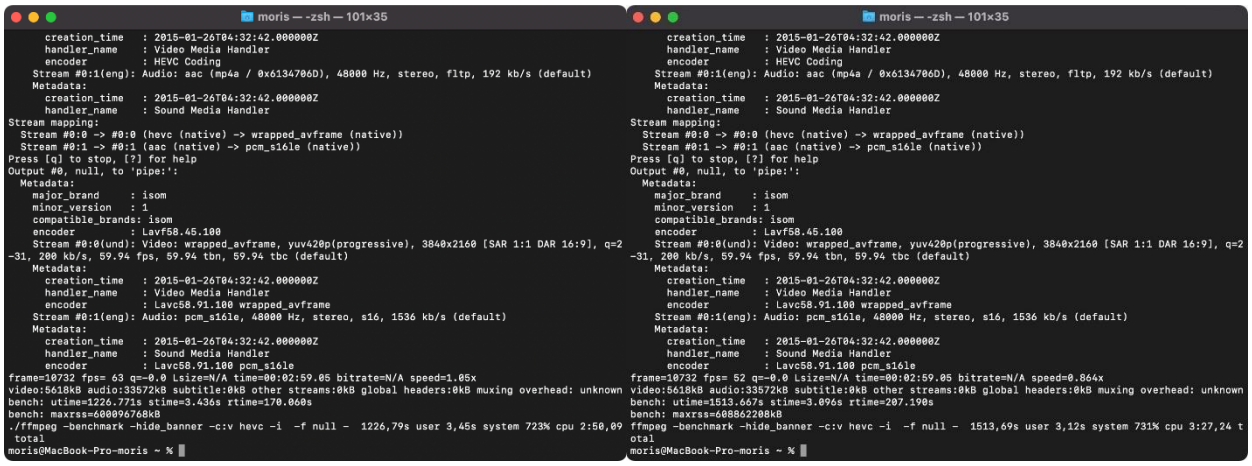
Выше прилагаю скриншот, поясняющий, почему мы не воспользуемся нашим PowerShell-скриптом для теста в ffmpeg. Тут причина в том, что PowerShell хоть и можно установить на macOS, но часть функций PowerShell на macOS недоступна, т.е. он тут не полноценный. В отличие от теста заполнения SSD, который я и проверял в PowerShell на macOS, скрипт для теста в ffmpeg не задействует недоступные в macOS функции, но я, на всякий случай, перестрахуюсь. Надобности в PowerShell то особо нет, я этот скрипт использую только упрощения проведения теста ради, а так-то я вполне могу и без скрипта тест в ffmpeg провести, потому мы так и поступим.

Далее вопрос: какой ffmpeg мы будем использовать? Мы можем установить его при помощи brew, можем загрузить сборку с OSXexperts.NET и можем загрузить с evermeet.cx. Есть ли разница? Да, есть.



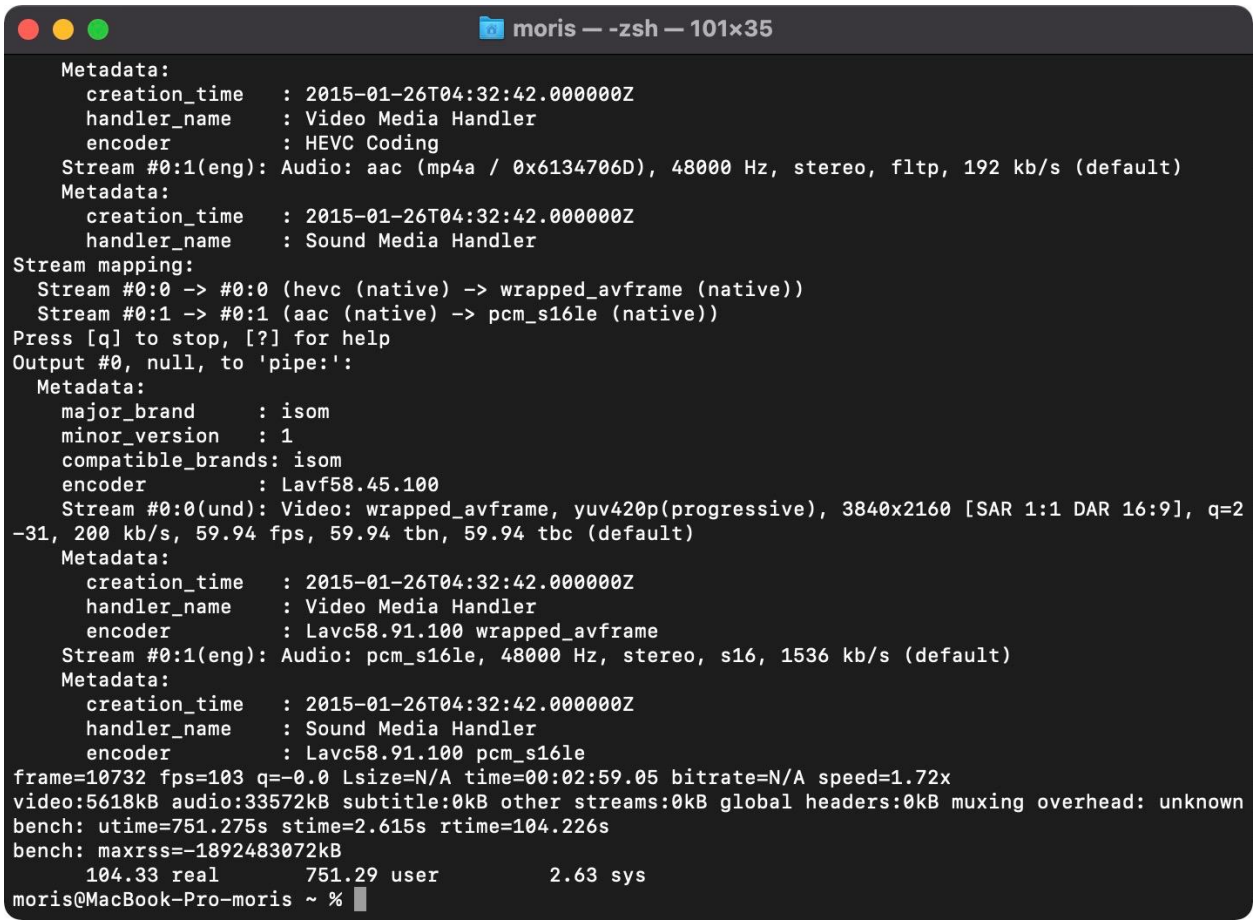
Скриншот 151. Сборка ffmpeg от osxexpers. Патч x265 NEON для ускорения кодирования видео в x265.

Сборка от OSXexperts.NET содержит патч ARM NEON, который в значительной мере повышает скорость кодирования в x265. Проверим.



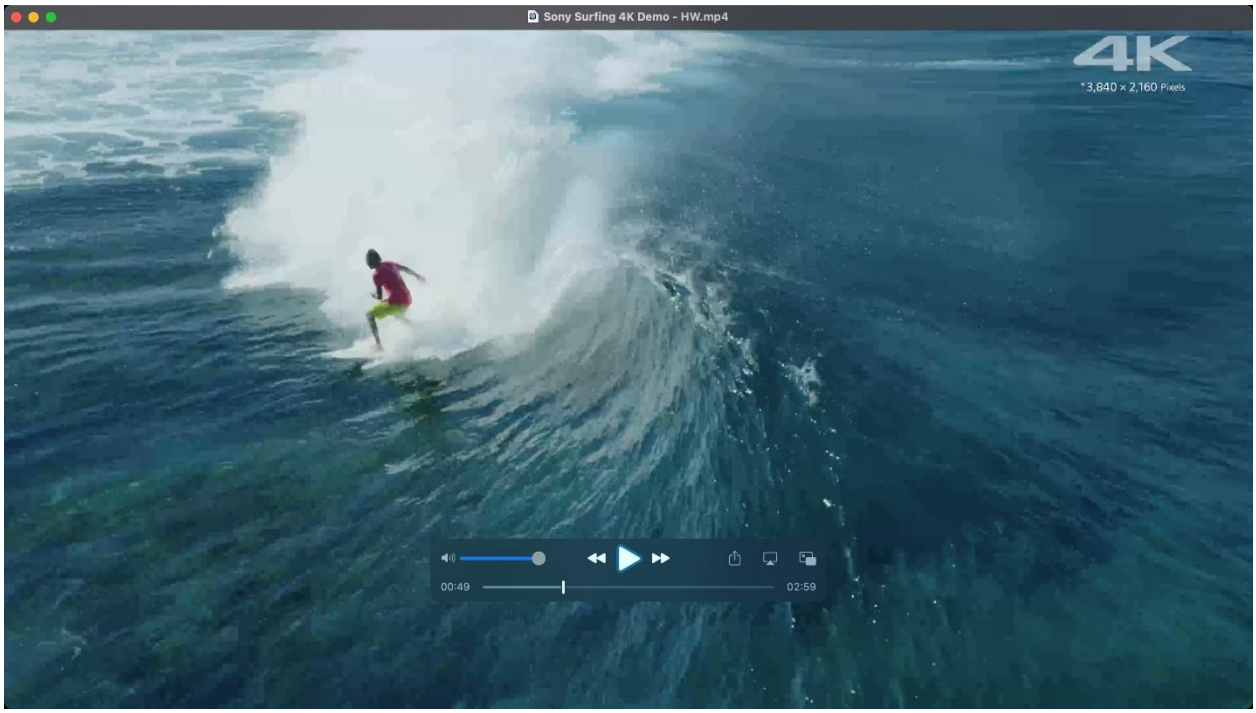
Скриншот 152. Macbook Pro (Apple M1). Скорость декодирования H.265 на CPU. Сравнение сборок OSXexperts и brew.

Я сравнил с версиями с evermeet.cx, и brew скорость декодирования, а не кодирования, т.к. мы в H.265 кодек не кодируем, а, наоборот, всегда перекодируем видео из H.265 в H.264 в наших тестах ноутбуков. И да, разница в этом сценарии есть, аж на 22% быстрее справилась сборка от OSXexperts, поэтому мы будем использовать её. Если же использовать блоки аппаратного ускорения, а не проводить декодирование силами центрального процессора, разницы между этими тремя сборками никакой нет.

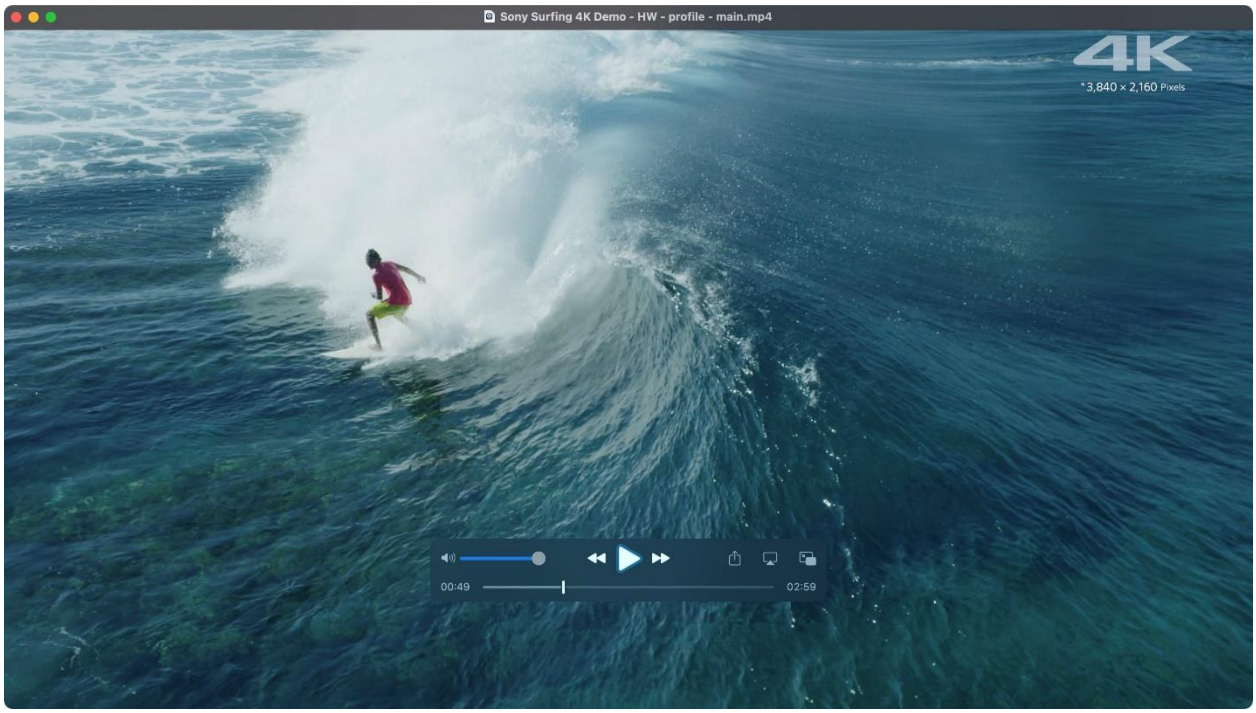


Скриншот 153. Macbook Pro (Apple M1). Скорость декодирования H.265 на CPU. Rosetta 2.

Я захотел сравнить результаты с использованием Rosetta 2 (сборки OSXexperts и evermeet.cx, у них результаты одинаковые) и я был немного ошарашен. При использовании сборки для x86 и Rosetta 2, скорость декодирования видео в кодеке H.265 вышла на 63% выше, чем скорость декодирования этого же самого видео со сборкой OSXexperts, а она, в свою очередь, была быстрее сборок brew и evermeet.cx. Что? Как это работает? Да, получается, бывают исключения, когда нативный машинный код под ARM выходит менее эффективным, чем транслированный при помощи Rosetta 2. И ffmpeg – одно из таких исключений. Впрочем, при использовании аппаратного ускорения, имеется небольшая разница в пользу нативного решения под ARM.



Скриншот 154. Качество картинки при кодировании видео аппаратными блоками без указания желаемого битрейта.



Скриншот 155. Качество картинки при кодировании видео аппаратными блоками с указанием желаемого битрейта.

Есть ещё один нюанс: ffmpeg при кодировании видео с использованием API Apple VideoToolBox по умолчанию вывел нам итоговое видео с битрейтом 7 Мбит вместо 104-111 Мбит, с которыми обычно выводят видео аналогичной командой аппаратные блоки Intel, AMD и NVidia. Поэтому в этот раз пришлось внести корректировку в команду для кодирования видео и добавить туда следующие параметры: **-profile:v main -b:v 110M**

По итогу наша команда для перекодирования видео аппаратными блоками по API VideoToolBox стала выглядеть так:

```
type="HW - profile - main"; time ./ffmpeg -benchmark -hide_banner -hwaccel
videotoolbox -c:v hevc -i ~/Desktop/Transcoding/Sony\ Surfing\ 4K\ Demo.mp4 -c:v
h264_videotoolbox -c:a copy -map 0:v:0 -map 0:a:0 -vsync 0 -qmin 18 -qmax 18 -
profile:v main -b:v 110M ~/Desktop/Transcoding/Sony\ Surfing\ 4K\ Demo\ -\
$type.mp4; ./ffmpeg -hide_banner -i ~/Desktop/Transcoding/Sony\ Surfing\ 4K\
Demo\ -\ $type.mp4 2>&1 | grep bitrate;
```

С тем, как мы проводим тест, разобрались, теперь взглянем на итог.

		CPU TDP (W)	GPU TDP (W)	Decode H.265 (sec)	Transcode H.265 -> H.264 (sec)	Output H.264 file size (MiB)	Output H.264 file bitrate (Mbit/s)
Asus ROG Zephyrus DUO **CPU**: Core i9 10980HK **GPU**: RTX 2080 Super Mobile	Intel QSV	70	90	50	144		
	NVIDIA CUVID			33	88		
	CPU Only (-crf 0)			72	1222		
Xiaomi Mi Notebook Pro GTX **CPU**: Core i7 8550U **GPU**: GTX 1050 Max-Q	Intel QSV	25	30	54	152		
	NVIDIA NVENC			115	153		
	CPU Only (-crf 0)			175	3701		
Xiaomi Mi Notebook Pro GTX - macOS 11.2 Big Sur **CPU**: Core i7 8550U **GPU**: UHD Graphics 620	VideoToolBox	25	15	115	215	144	7
	CPU only			139	2235	3008	141
	CPU only (-crf 0)			139	3160	32160	1507
RedmiBook 16 2020 **CPU**: Ryzen 7 4700U **GPU**: Radeon Vega 8	AMF (D3D11VA)	25	19	99	279	2223	104
	CPU Only			143	1267	3003	141
	CPU Only (-crf 0)			143	1822	32073	1502
Acer Nitro 5 AN515-44 **CPU**: Ryzen 5 4600H **GPU**: GeForce GTX 1650	AMF (D3D11VA)	35	50	71	186		
	NVIDIA CUVID			51	92		
	CPU Only			86	1160		
	CPU Only (-crf 0)			86	1590		
Dell XPS 17 9700-6727 **CPU**: Core i7 10875H **GPU**: GeForce RTX 2060 Max-Q	Intel QSV	55	65	52	170	2329	109
	Nvidia CUVID			47	100	2363	111
	D3D11VA			193	1000	3006	141
	CPU Only			83	1039	3006	141
	CPU Only (-crf 0)			83	1379	32162	1507
Asus ROG Strix Scar 17 G732LXS **CPU**: Core i9 10980HK **GPU**: GeForce RTX 2080	Intel QSV	90	150	49	165	2329	109
	Nvidia CUVID			42	84	2363	111
	D3D11VA			175	903	3006	141
	DXVA2 (D3D9VA)			63	891	3006	141
	CPU Only			74	927	3006	141
	CPU Only (-crf 0)			74	1240	32162	1507
Asus ZenBook Flip UX363EA **CPU**: Core i7 1165G7 **GPU**: Intel Iris Xe	Intel QSV	15	10	32	149	2333	109
	D3D11VA			95	2435	3003	141
	DXVA2 (D3D9VA)			87	2412	3003	141
	CPU Only			181	2527	3003	141
	CPU Only (-crf 0)			181	3646	32154	1506
HP Omen 15 15-en0037ur **CPU**: Ryzen 5 4600H **GPU**: GTX 1660 Ti	AMF (D3D11VA)	53	80	95	201	2372	111
	AMF (DXVA2)			104	208	2372	111
	NVidia CUVID			52	92	2367	111
	CPU Only			81	974	3009	141

	CPU Only (-crf 0)			81	1317	32160	1507
Asus ROG Strix G15 **CPU**: Ryzen 9 5900HX **GPU**: GeForce RTX 3070 Mobile	AMF (D3D11VA)	80	130	83	500	2372	111
	AMF (DXVA2)			94	533	2372	111
	NVidia CUVID			43	90	2329	109
	CPU Only			64	699	3010	141
	CPU Only (-crf 0)			64	877	32167	1507
Lenovo Yoga Slim 7 **CPU**: Ryzen 7 4800U **GPU**: Radeon RX Vega 8	AMF (D3D11VA) Performance	25	15	95	247	2372	111
	AMF (D3D11VA) Intellig. Cooling	20	15	97	250	2372	111
	AMF (DXVA2) Performance	25	15	96	248	2372	111
	AMF (DXVA2) Intellig. Cooling	20	15	98	250	2372	111
	CPU Only Performance	25	15	84	986	3010	141
	CPU Only Intellig. Cooling	20	15	98	1249	3010	141
	CPU Only (-crf 0) M. Performance	25	15	84	1330	32167	1507
	CPU Only (-crf 0) Intellig. Cooling	20	15	98	1663	32167	1507
Apple Macbook Pro 2020 Z11B0004T **CPU**: Apple M1 **GPU**: Apple M1 8 cores	VideoToolBox	20	10	143	339	160	7
	VideoToolBox Rosetta 2			159	337	160	7
	VideoToolBox manual bitrate			145	348	2336	109
	VideoToolBox manual bitrate Rosetta 2			163	346	2336	109
	CPU only			170	1403	4784	224
	CPU only Rosetta 2			104	2468	4784	224
	CPU only (-crf 0)			170	1738	32160	1507
	CPU only (-crf 0) Rosetta 2			104	2498	32160	1507

Таблица 4. Сравнительная таблица с результатами тестирования скорости перекодирования в ffmpeg.

Совершенно ничего выдающегося не вижу. Может быть, ffmpeg ещё не адаптирован в должной мере для работы с API Apple VideoToolBox, тогда всё в целом понятно. Иначе как объяснить, что ffmpeg даже на моём Xiaomi на Windows при задействовании Intel Quick Sync Video справляется с задачей декодирования в 2 раза быстрее, чем он же, но на macOS 11.2 Big Sur при задействовании API VideoToolBox? Я просто не могу винить Apple в том, что тут нисколько не удивительные результаты, т.к. очевидно, что проблема именно в том, как ffmpeg задействует API VideoToolBox, поэтому мы перейдём к следующему тесту. Хотя, он в принципе не относится к тестам блоков аппаратного декодирования, это скорее тест CPU, ну да ладно.

7.3. Тесты с кодеком AV1 при помощи dav1d.

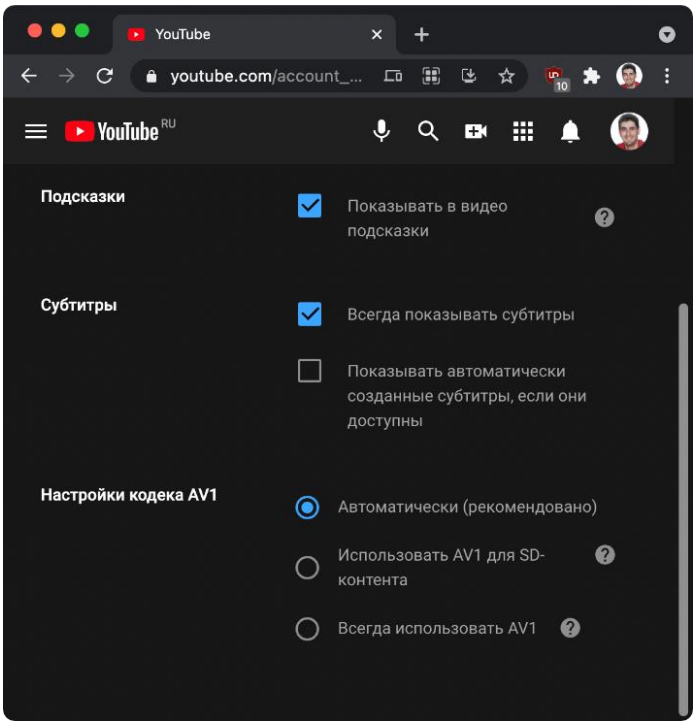
Если Вы на днях уже читали обзор на Lenovo Yoga Slim 7, для Вас этот тест не станет сюрпризом. Не так давно, я наткнулся в твиттере на один твит VideoLAN, разработчиков плеера VLC, где они сравнивают 2 яблочных аппарата, показывая насколько Apple M1 ушёл вперёд от i9-9980HK. И мне стало любопытно повторить этот тест и взглянуть на результаты у себя.

В чём суть: на сегодняшний день ни один распространённый декодер видео AV1 не умеет задействовать аппаратные блоки декодирования видео в SoC для аппаратов, работающих под управлением ОС Windows.

Есть всего 2 распространённых решения: LIBGAV1 от Google и DAV1D от VideoLAN (разработчики плеера VLC). На сайте phoronix было опубликовано сравнение по части производительности этих двух решений, и, по результатам тестов phoronix, разработка Google осталась далеко позади.

Чего уж там, Google теперь и сами используют DAV1D в Chrome, отказавшись от своего предыдущего детища LIBGAV1. Наряду с Google, в своём браузере Firefox по умолчанию DAV1D задействовали и Mozilla.

И мне любопытно, а с какой частотой кадров Macbook Pro с Apple M1 способен декодировать видео в кодеке AV1, который сейчас даже YouTube стал использовать по умолчанию. Откуда я это знаю?



Скриншот 156. Настройки кодека AV1 на странице настроек воспроизведения YouTube.

Перейдите на страницу настроек проигрывателя YouTube. Если Ваш браузер поддерживает декодирование видео с кодеком AV1, Вы увидите возможность отключения AV1 для разрешения выше 480p (вариант «Использовать AV1 для SD-контента»). По умолчанию лично у меня на моём Xiaomi Mi Notebook Pro GTX AV1 почему-то используется для всех разрешений, и точно такую же ситуацию я наблюдал на ноутбуках HP Omen 15, Asus ROG Strix G15 и Lenovo Yoga Slim 7. И это странно. Зачем на устройства, которые аппаратно не поддерживают декодирование кодека AV1, транслировать видео в этом кодеке по умолчанию, когда есть ещё кодеки VP9 и H.264, аппаратное декодирование которых поддерживается ими?

Более того, самое абсурдное во всей это ситуации, так это то, что декодер AV1 видео DAV1D не поддерживает аппаратное декодирование ни на одном аппарате, он декодирует кодек AV1 на ЦП даже в SoC Apple, Mediatek, Samsung и Intel, имеющих отдельные блоки для декодирования.

В чём вообще смысл трансляции видео в AV1 поголовно на все аппараты, если его аппаратное декодирование на данный момент не представляется возможным ни в одном браузере?

Ну да ладно, раз уж декодер DAV1D так распространён, мы его протестируем и посмотрим на то, как быстро Apple M1 справится с декодированием 3 видео в кодеке AV1.

Декодер мы уже скомпилировали из исходного кода в главе 5.9, и готовые для запуска исполняемые файлы (программы, короче говоря) я опубликовал в своём форке DAV1D на GitHub. Перед запуском на macOS введите в терминале команду: sudo chmod +x ./tools/dav1d

Предупрежу, что для Windows я dav1d собирал не нативно с самой ОС Windows, а через Ubuntu в WSL2 (Windows Subsystem for Linux), т.к. нативно мне собрать dav1d из исходного кода не удалось. Результаты на Windows могут быть несколько хуже из-за отсутствия должной оптимизации при кросс-компиляции под Windows из-под Linux, но лично я особой разницы не заметил.

Тест проведём на 3 сэмплах от Netflix:

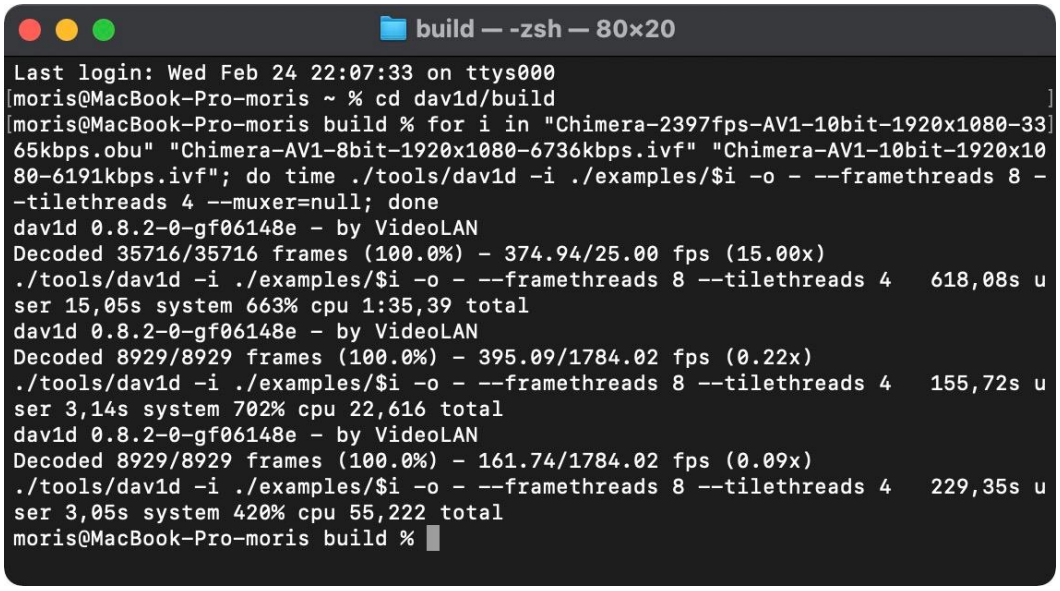
2. Chimera AV1 1080p 23.976 FPS 10-bit 3365 kbit/sec (obu).
3. Chimera AV1 1080p 23.976 FPS 8-bit 6736 kbit/sec (ivf).
4. Chimera AV1 1080p 23.976 FPS 10-bit 6191 kbit/sec (ivf).

Кладём сэмлы в папку examples в dav1d и выполняем след. команду:

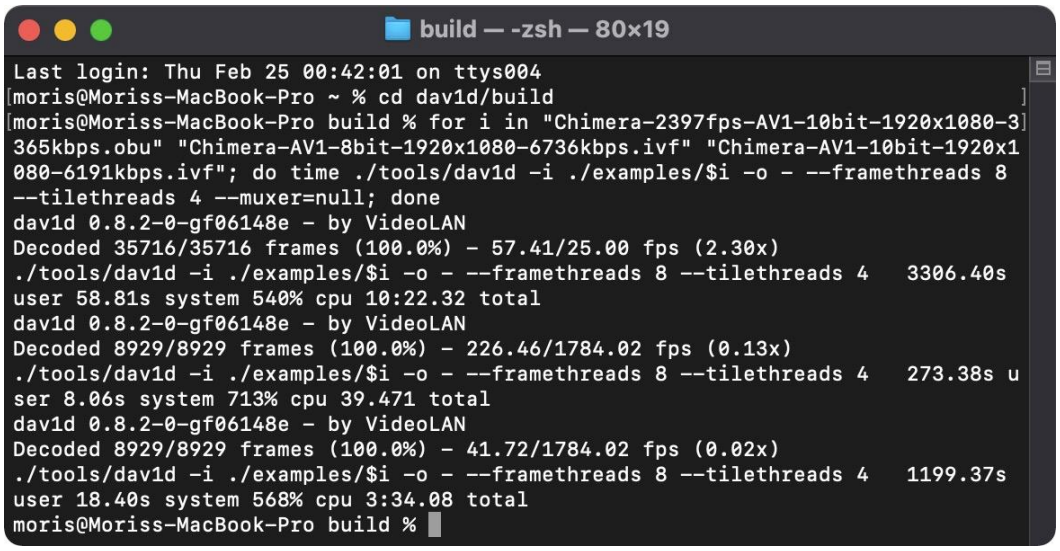
```
for i in "Chimera-2397fps-AV1-10bit-1920x1080-3365kbps.obu" "Chimera-AV1-8bit-1920x1080-6736kbps.ivf" "Chimera-AV1-10bit-1920x1080-6191kbps.ivf"; do time ./tools/dav1d -i ./examples/$i -o - --framethreads 8 --tilethreads 4 --muxer=null; done
```

Аналогичная команда для PowerShell в Windows:

```
foreach ($testvideo in "Chimera-2397fps-AV1-10bit-1920x1080-3365kbps.obu","Chimera-AV1-8bit-1920x1080-6736kbps.ivf","Chimera-AV1-10bit-1920x1080-6191kbps.ivf") {Measure-Command -Expression {.\tools\dav1d.exe -i .\examples\$testvideo -o - --framethreads 8 --tilethreads 4 --muxer=null} | findstr TotalSeconds}
```

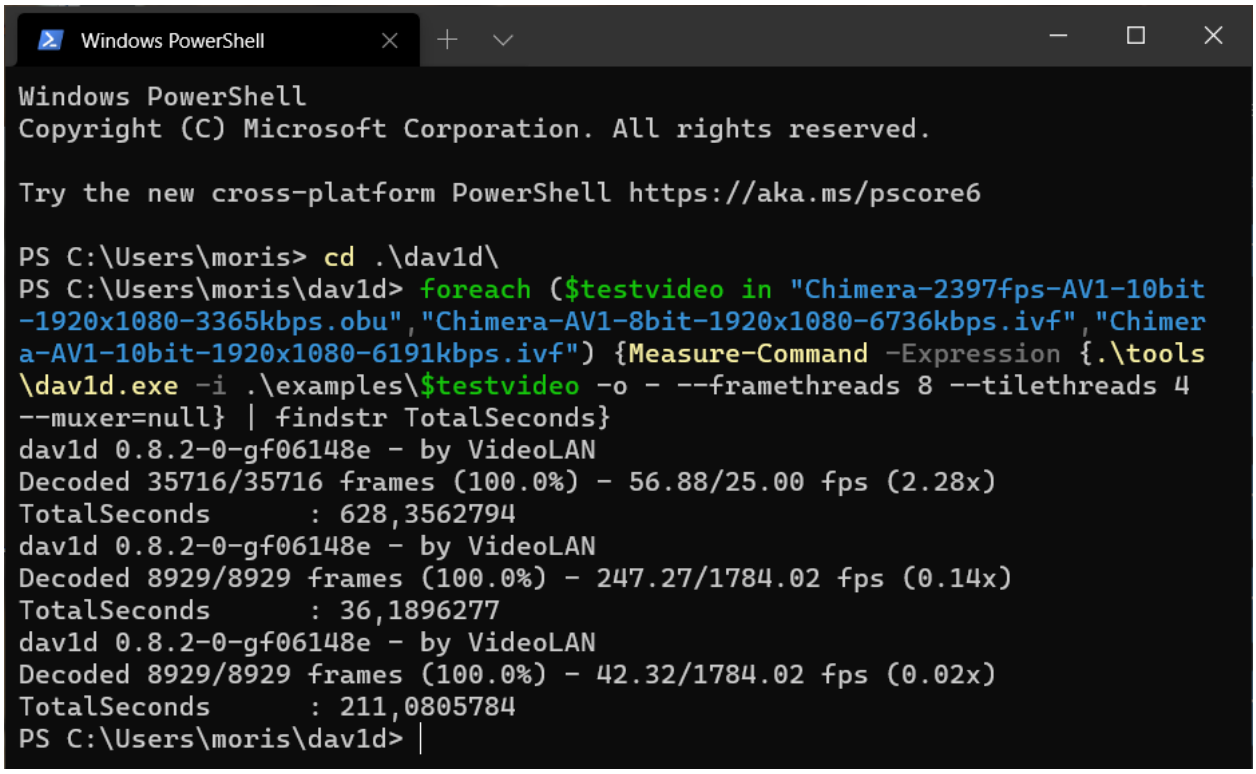


Скриншот 157. Macbook Pro 13 2020 (Apple M1, 20W). Скорость декодирования AV1 видео в DAV1D.



Скриншот 158. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). macOS 11.2. Скорость декодирования AV1 видео в DAV1D.

Это просто уничтожение. То, что i7 8550U в моём Xiaomi выполнял более 10 минут, Apple M1 просчитал за 1.5 минуты. Что? Как?! Это разница в 6.5 раз для 1-го сэмпла, 1.75 раза для 2-го и 3.88 раза для 3-его. Ещё раз: как?!



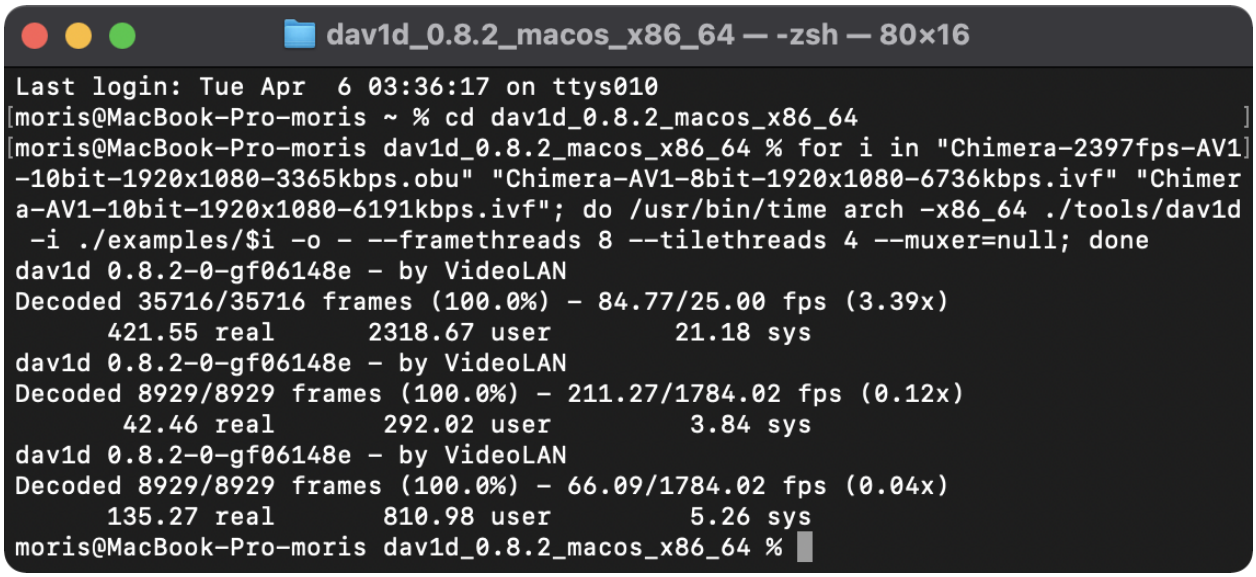
Скриншот 159. Xiaomi Mi Notebook Pro GTX (i7 8550U, 25W). Windows 10. Скорость декодирования AV1 видео в DAV1D.

Я провёл этот же тест на своём Xiaomi и на Windows 10, чтобы убедиться, что это не какой-то косяк скомпилированной под macOS версии, но нет, на Windows 10 версии 1909 я получил схожие результаты с теми, что на хакинтоше (macOS 11.2). Предвещая вопросы «а почему не обновил винду до 20H2?», отвечаю: мой Xiaomi ловит синий экран смерти при обновлении.

Мы провели этот же тест и на Lenovo Yoga Slim 7, который был у нас под рукой (я на него обзор параллельно с Macbook Pro на Apple M1 готовил, как и параллельно с Asus ROG Strix G15), и на декодирование 1-го сэмпла он затратил в 4,44 раза больше времени, чем Macbook Pro 13 с Apple M1, а на декодирование 2-го и 3-его сэмплов – в 1.1 и 2.57 раза больше времени соответственно. В этом тесте Macbook Pro с Apple M1 просто не оставляет шансов своим конкурентам (за исключением 2-го сэмпла, он меньше чем за 40 секунд декодируется на всех аппаратах), при том, что результаты Lenovo я привёл в режиме максимальной производительности, когда тот в среднем потребляет 25 Ватт после 3 минут нагрузки и 30-32 Ватта в первые 3 минуты.

Я подумал, что такого просто быть не может и решил провести ещё тест с Rosetta 2, не зря я ведь скомпилировал dav1d и из под x86. Чуть подправил команду для запуска x86 варианта через Rosetta 2:

```
for i in "Chimera-2397fps-AV1-10bit-1920x1080-3365kbps.obu" "Chimera-AV1-8bit-1920x1080-6736kbps.ivf" "Chimera-AV1-10bit-1920x1080-6191kbps.ivf"; do /usr/bin/time arch -x86_64 ./tools/dav1d -i ./examples/$i -o - --framethreads 8 --tilethreads 4 --muxer=null; done
```



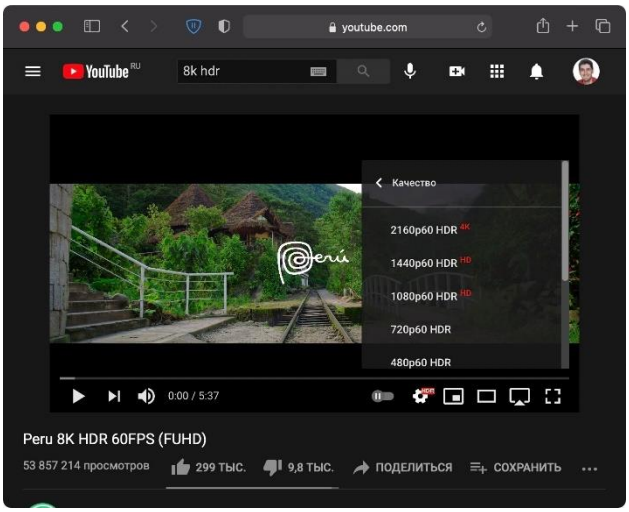
Скриншот 160. Macbook Pro 13 2020 (Apple M1, 20W). Скорость декодирования AV1 видео в DAV1D. Rosetta 2.

Разумеется, результаты по скорости декодирования AV1 кодека сильно зависят от битрейта декодируемого видео, от глубины цвета и других факторов, Вы и сами видели, что в зависимости от сэмпла разница менялась кардинальным образом, и всё равно результаты просто умопомрачительные. Я повторял тесты с Rosetta 2 и без несколько раз, думая, что это мог быть разовый глюк, но разница всегда оставалась в пределах погрешности между результатами с Rosetta 2, и точно также результаты без Rosetta 2 между собой не особо разнились. В целом выходит так, что Rosetta 2 довольно сильно сказывается на производительности при декодировании видео в кодеке AV1, разница достигает аж 4.42 раза, что никак не соотносится с теми 20-30% разницы с Rosetta 2 и без, которые мы наблюдали в предыдущих тестах процессора. С Rosetta 2 Macbook Pro с Apple M1 на борту уже практически сравнялся с Lenovo Yoga Slim 7 с Ryzen 7 4800U по времени рендеринга 1-го и 3-го сэмлов, а на 2-ом сэмпле он отстал даже от моего Xiaomi. Но без Rosetta 2 он так быстр, что ему ни один из этих двух не ровня.

7.4. Тесты декодирования 8K и 4K видео в YouTube.

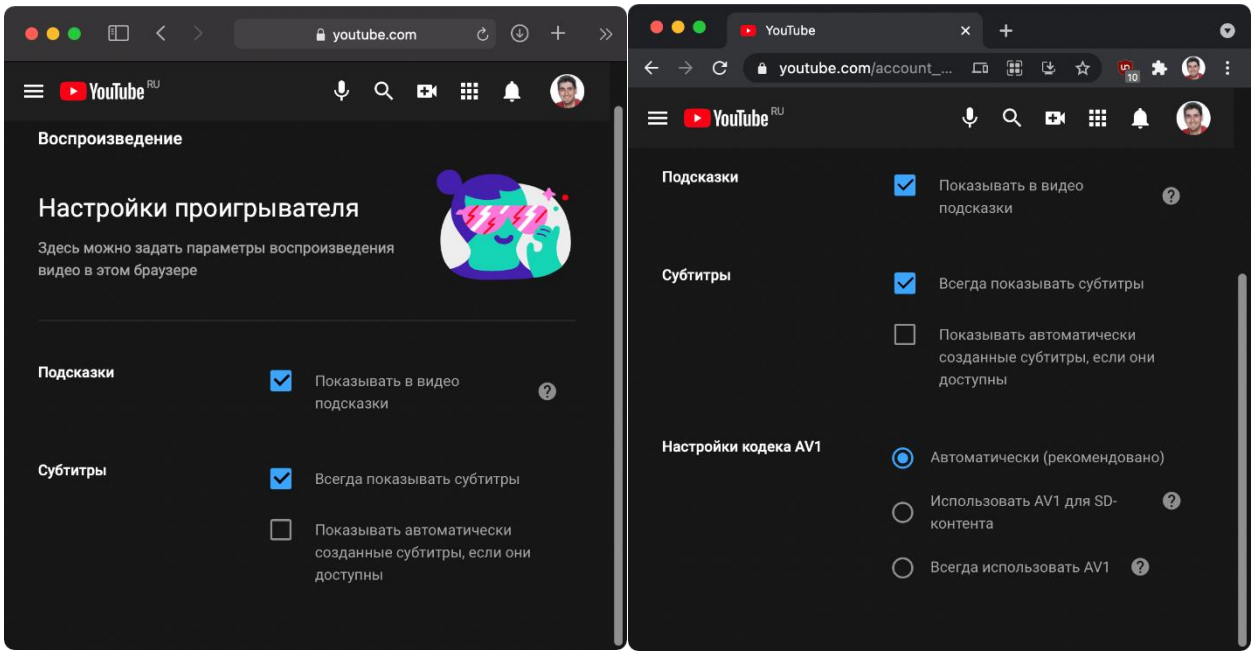
Ну, тут всё просто: находим 4 видеоролика в 8K, которые удовлетворяют нашим критериям, а затем проверяем с помощью ***stats for nerds*** на YouTube то, сколько кадров у нас было потеряно. Однако есть нюансы, ну куда же без них. Нет, сэмплы то мы нашли подходящие:

1. AV1 8K HDR 60 FPS – Peru.
2. AV1 8K HDR 30 FPS – Bali.
3. AV1 8K SDR 60 FPS – Japan.
4. VP9 8K SDR 30 FPS – Beauties of the World.



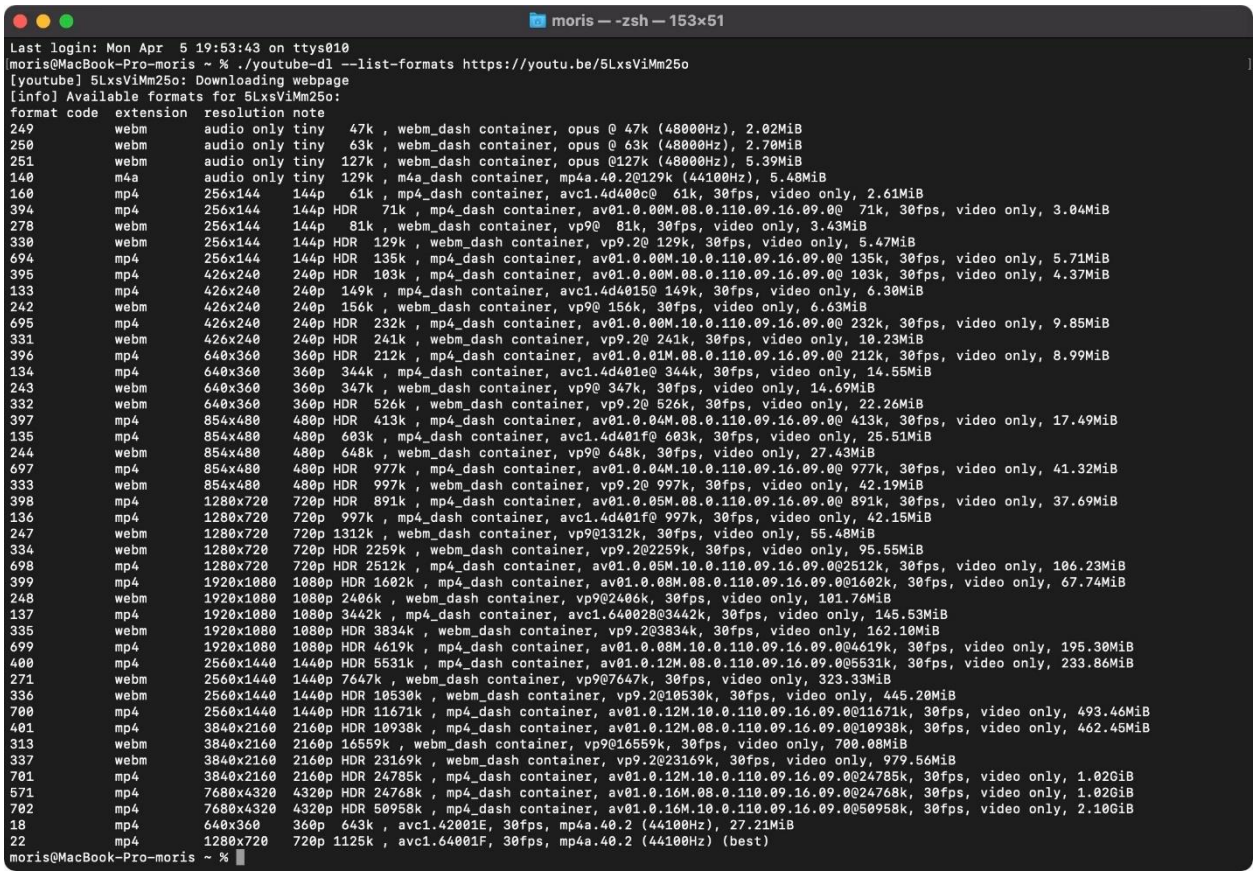
Скриншот 161. Safari. Невозможность воспроизведения 8K видео в 8K.

Safari, в отличие от Chrome и Firefox, не позволял мне воспроизвести 3 из 4 сэмплов в 8K. В чём же причина? Она оказалась весьма проста.



Скриншот 162. Страница настроек воспроизведения видео в YouTube. Safari (слева) и Chrome (справа).

Дело в том, что Safari попросту не поддерживает воспроизведение видео в кодеке AV1. На странице настроек воспроизведения видео настройки кодека AV1 отсутствуют, тогда как в других браузерах присутствуют.



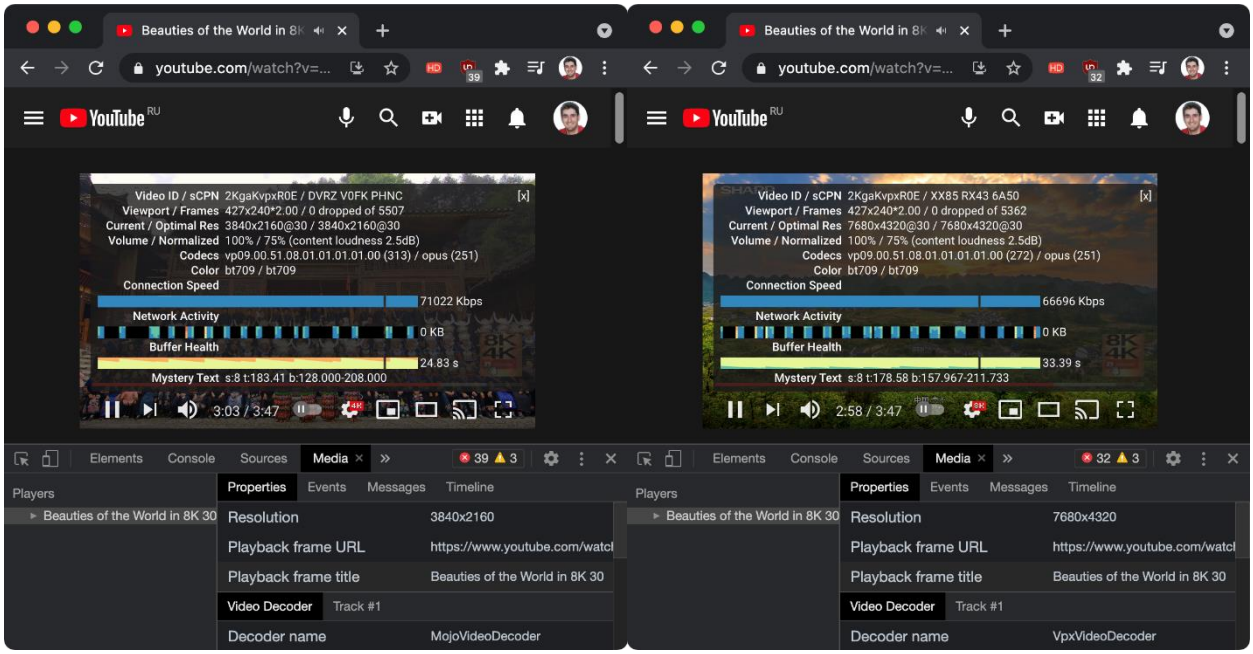
Скриншот 163. Доступные форматы для видео Bali (8K AV1 HDR 30 FPS).

Проверить то, в каких кодеках YouTube может транслировать Вам видео, и в каких кодеках Вы можете загрузить это самое видео, Вы можете при помощи консольной утилиты youtube-dl. Дело в том, что все SDR видео, которые мы нашли в 8K с частотой кадров 60 FPS, а также все HDR видео в 8K транслируются YouTube`ом Вам в кодеке AV1 (выше обозначен как ***av01***). Соответственно, если кодек AV1 Ваш браузер не поддерживает, вместо видео в 8K Вы увидите видео в максимальном разрешении, поддерживаемого Вашим браузером кодека (VP9 или H.264). В VP9 HDR (8K 30 или 60 FPS) видео можно посмотреть максимум в 4K, как и SDR 60 FPS видео, а H.264 (выше на скриншоте обозначен как ***avc1***) YouTube транслирует Вам в разрешении не выше 1080p.

Т.е. для того, чтобы протестировать 3 из 4 видео, нам потребуется браузер Chrome или Firefox, но не Firefox. И тут встаёт проблема: да, да,

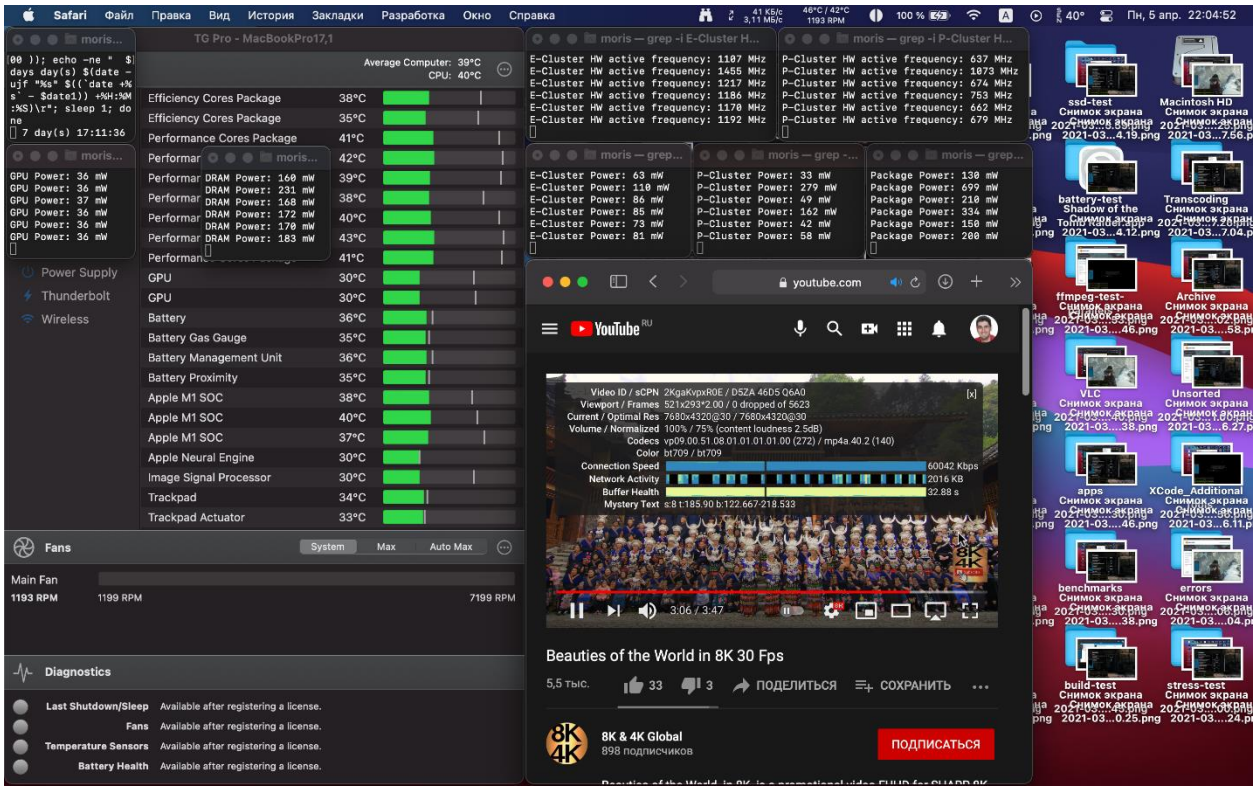
именно то, что я писал выше в главе 7.3: ни один сегодняшний браузер не поддерживает аппаратное декодирование видео в кодеке AV1. Несмотря на то, что нужные блоки аппаратного декодирования присутствуют в SoC Apple M1, отсутствие поддержки этих самых блоков декодером DAV1D, который используется по умолчанию в Chrome и Firefox при декодировании AV1 видео, означает, что декодирование 8K видео в 3 сэмплах из 4 будет выполняться на процессоре, и только 1 сэмпл мы можем протестировать с кодеком VP9, чтобы понять, могут ли в принципе блоки аппаратного декодирования видео Apple декодировать это самое видео хоть в одном кодеке в 8K?

Хотя, вообще, отличный вопрос компании Apple: почему Ваш Safari на Вашем же ноутбуке, SoC которого поддерживает аппаратное декодирование видео в кодеке AV1, не позволяет воспроизводить это самое видео в AV1? В чём смысл пихать эти блоки и не задействовать их самим никак?



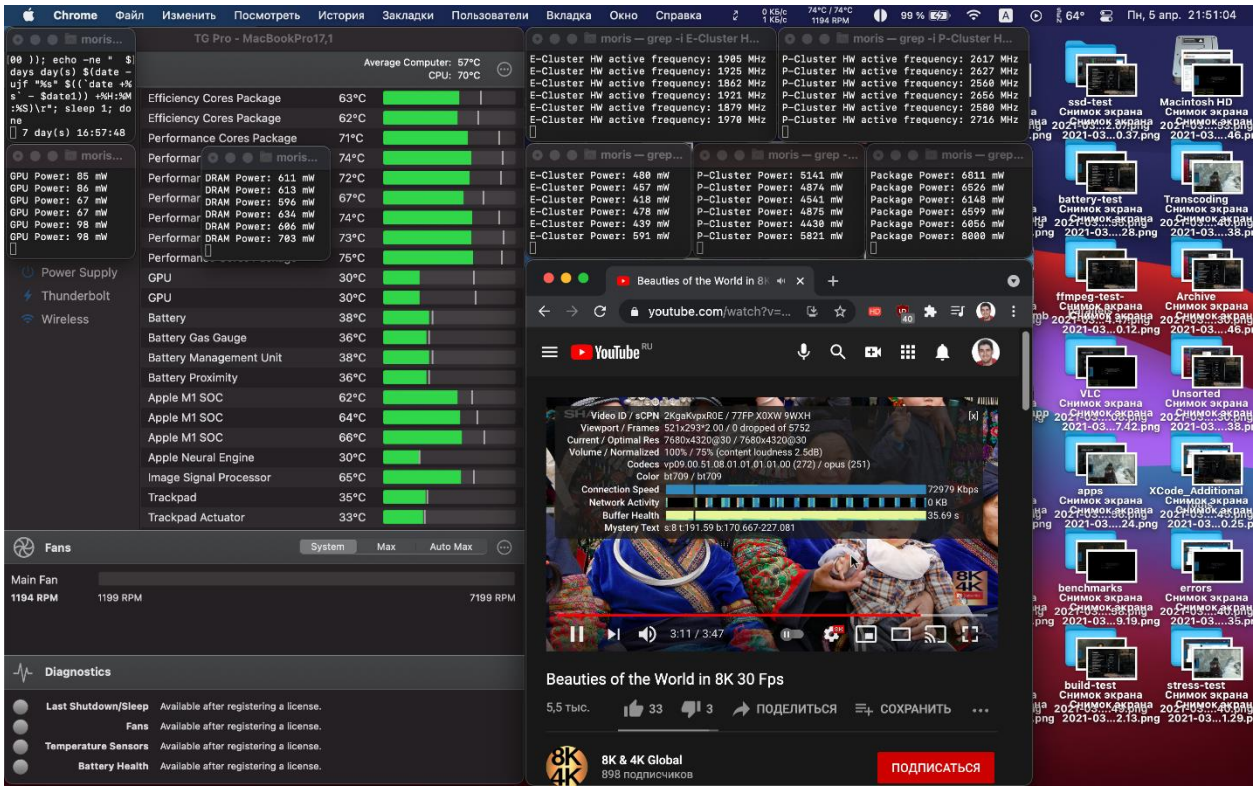
Скриншот 164. Используемый декодер в Chrome при воспроизведении VP9 SDR 30 FPS видео: 4K и 8K.

Но и тут всё оказалось не так гладко, как хотелось бы. Дело в том, что Chrome использует декодер ***VpxVideoDecoder*** при воспроизведении 8K видео. ***VpxVideoDecoder***, в отличие от ***MojoVideoDecoder***, который Chrome задействует при воспроизведении видео в 4K, не поддерживает аппаратное ускорение VP9. И что в итоге, значит, M1 не имеет блоков аппаратного ускорения видео в кодеке VP9 и в разрешении 8K? А вот и нет.



Скриншот 165. Воспроизведение 8K30 SDR видео в Safari.

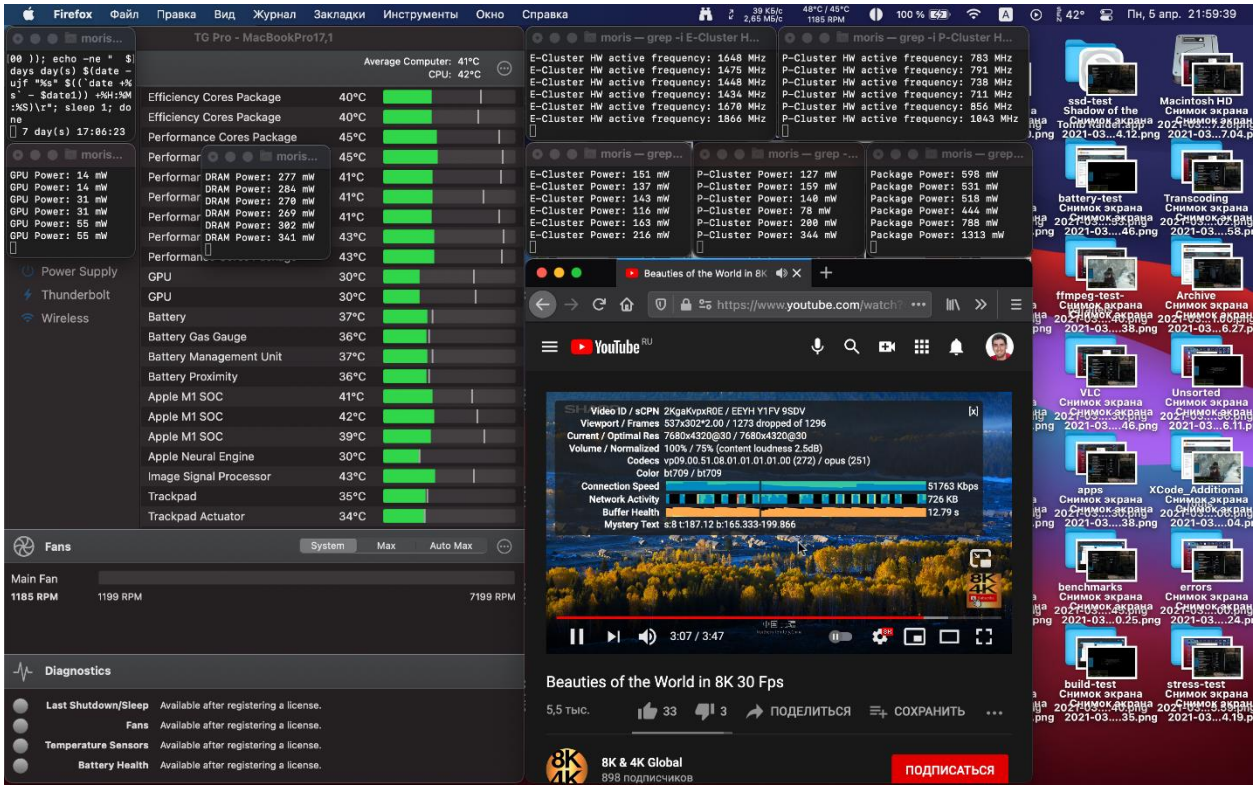
Safari не поддерживает AV1, зато 8K видео в кодеке VP9 он без труда декодирует аппаратными блоками, при использовании которых общее энергопотребление SoC составляет в среднем лишь 200-300 мВ. Это сильно.



Скриншот 166. Воспроизведение 8K30 SDR видео в Chrome.

Для сравнения, Chrome при декодировании того же самого видео в том же разрешении 8K потребляет 6-8 Ватт энергии, т.к. эту задачу выполняют не отдельные блоки аппаратного ускорения, наиболее подходящие для данной задачи, а центральные ядра процессора и у них потребление энергии выше.

Видать, энергоэффективность - новый конёк Apple. 200-300 мВ... С ума сойти можно. Я не видел ранее SoC, который в схожей задаче потреблял бы сопоставимое количество энергии, у всех остальных оно было сильно выше.

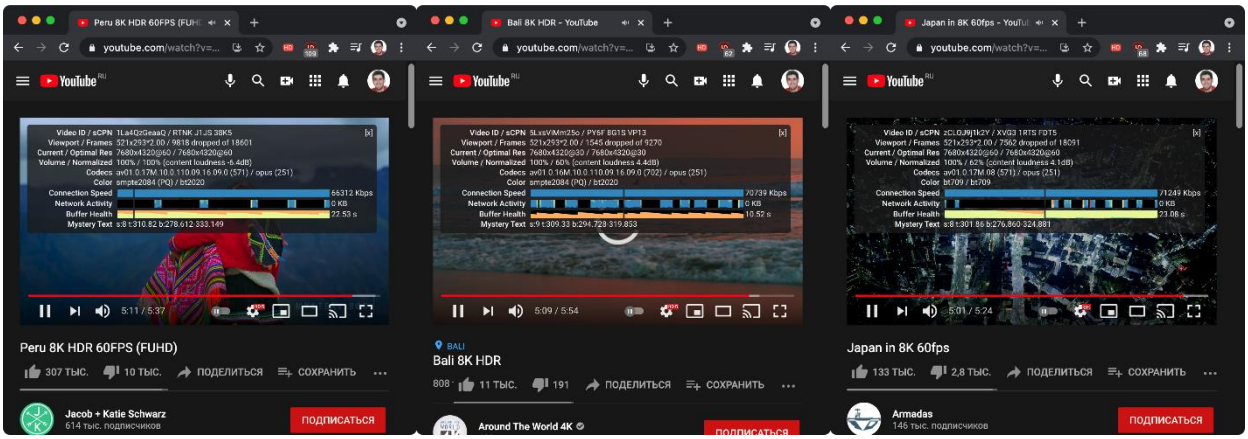


Скриншот 167. Воспроизведение 8K30 SDR видео в Firefox.

Про Firefox, к сожалению, говорить не приходится, т.к. при воспроизведении 8K видео он пропустил 98% кадров по статистике YouTube. Так-то 98% — это мягко сказано, т.к. на деле перед глазами за 3.75 минуты я видел лишь 4 статичных кадра.... В общем, Firefox с воспроизведением 8K видео, даже в 30 FPS, даже в SDR, не справляется. С данной задачей лучше всех справляется Safari, а чуть хуже - Chrome.

К слову, информацию о кодеке Вы можете увидеть в stats for nerds на YouTube: ***vp09*** в поле Codecs значит кодек VP9, а ***av01*** значит кодек AV1.

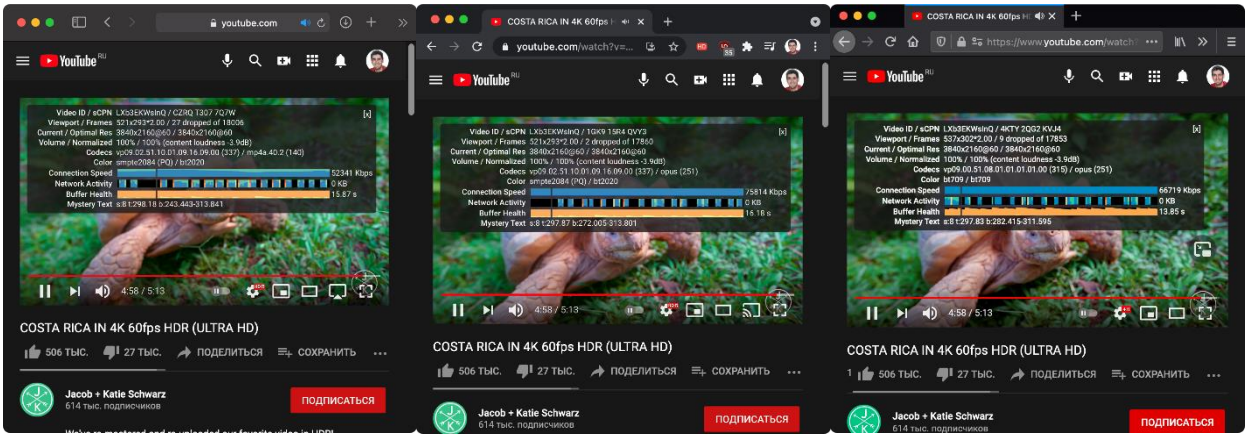
А как дела у Chrome с воспроизведением 8K 60 FPS и 8K HDR видео?



Забудьте, Chrome также с воспроизведением 8K AV1 видео не справляется, будь то 8K AV1 SDR 60 FPS, или же 8K AV1 HDR (30 или 60 FPS), без разницы, в любом случае наблюдаются пропуски кадров до 53%.

Может быть, даже и не плохо, что Apple ещё не реализовали декодирование AV1 видео в своём браузере Safari. Лучше уж дождаться нормальной реализации, задействующей аппаратные блоки декодирования.

А как там дела с 4K?



Скриншот 168. Воспроизведение 4K HDR 60 FPS видео в Safari (слева), Chrome (центр) и Firefox (справа).

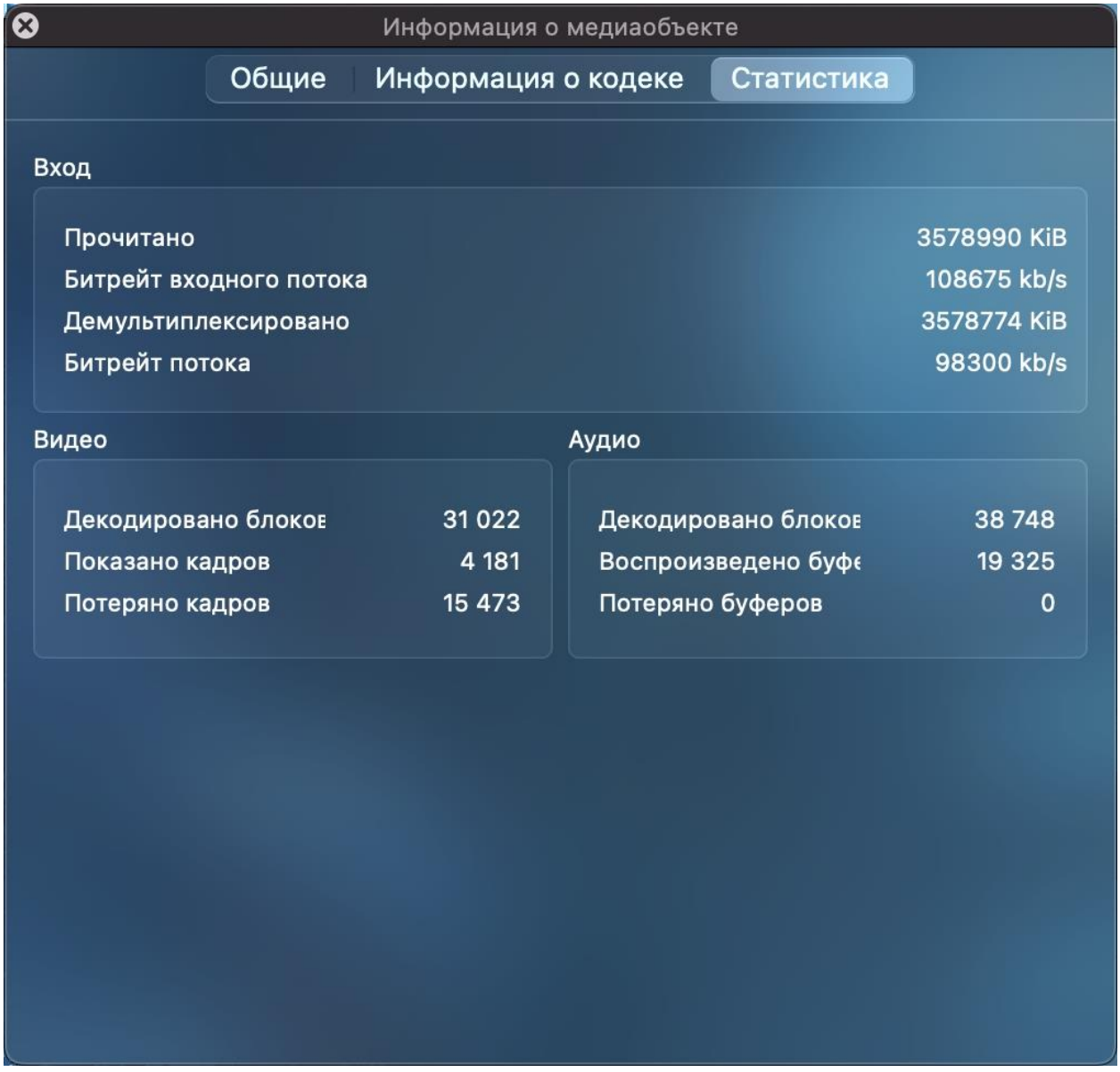
С 4K HDR 60 FPS видео никаких проблем не возникло у Safari и Chrome, а вот Firefox, как оказалось, воспроизведение HDR видео не поддерживает, поэтому в графе Color у него и bt709 вместо bt2020.

В целом у Mac с Apple M1 нет проблем с 4K видео, а вот 8K 30 FPS SDR видео пока нормально воспроизводится лишь в Safari.

Умеют ли блоки аппаратного ускорения воспроизводить видео в разрешении 8K? Да, умеют. С каким кодеком? Через Safari кодек VP9 спокойно обрабатывается блоками аппаратного ускорения, но без HDR. Может быть, поддержка HDR у самих блоков есть, но её нет у Safari?

7.5. Тесты 8K и 4K видео в VLC.

В общем-то найти нормальные сэмплы в 8K – дело не столь простое, сколь может показаться на первый взгляд. В кодеке H.265/HEVC я нашёл 8K 60 FPS сэмплы лишь на одном сайте на весь интернет: joumxyzptlk.de. Я выбрал для загрузки вариант «3DMark 2013 Fire Strike Ultra recorded in 7680x4320 60 fps (unhacked blurry)» в разрешении 7680x4320 (60 FPS). Стандартное, казалось бы, 8K разрешение, глубина цвета 8-bit и 60 FPS. Что же могло пойти не так?



Скриншот 169. Число потерянных кадров при воспроизведении 8K H.265 8-bit 60 FPS видео в VLC.

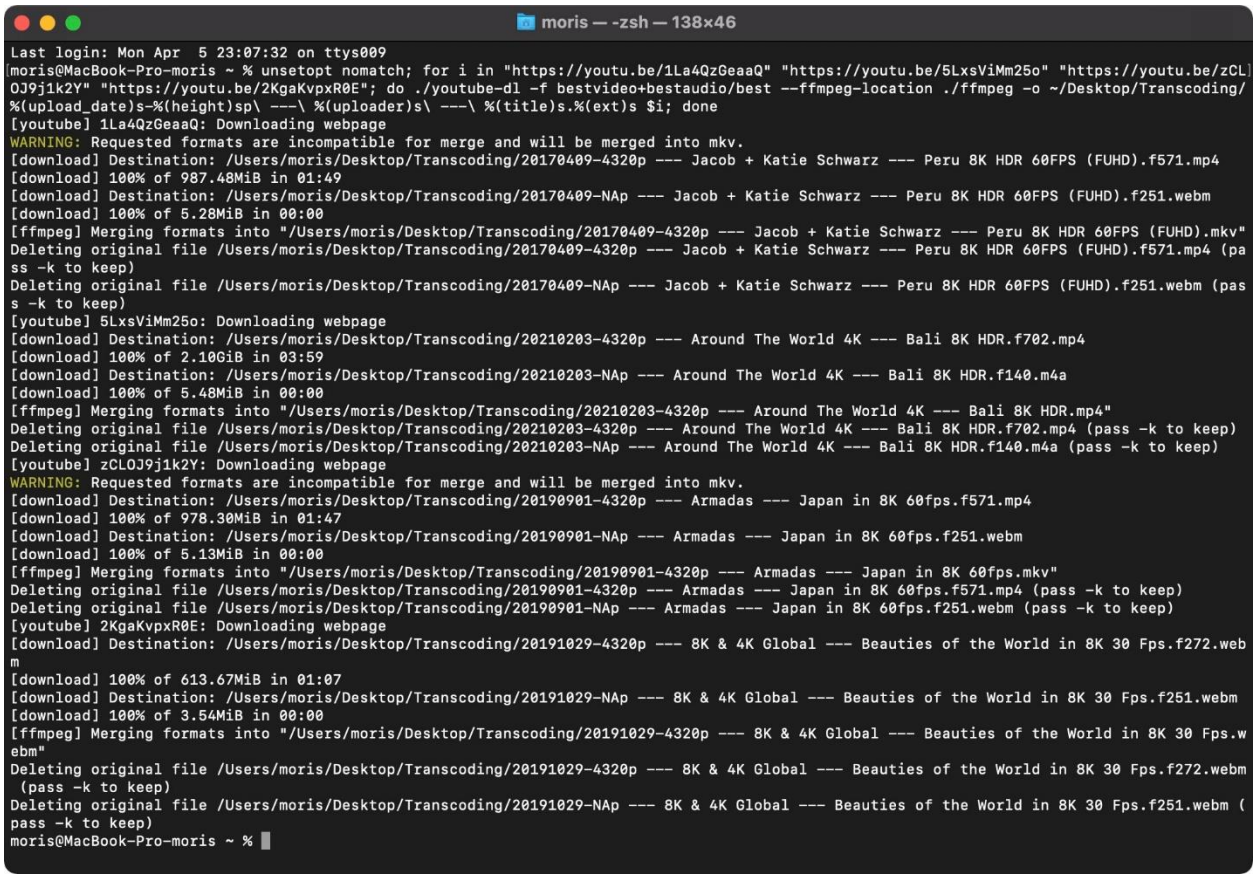
Число потерянных кадров – 15473 из общего числа в 19654 кадра, т.е. всего кадров было пропущено 79%. Более того, я бы и числу показанных кадров верить не советовал, т.е. я в видео увидел лишь 5 статичных кадров.

Как мне найти другие 8K сэмплы с кодеками VP9 и AV1? Идея пришла на ум простейшая: скачать 8K видео с YouTube при помощи youtube-dl.

При помощи следующего набора команд я загрузил себе все 8K видео, которые я проверял выше в главе 7.4 (**UPDATE**: вместо youtube-dl лучше используйте yt-dlp, т.к. он чаще обновляется, он своевременно адаптируется под изменения YouTube и с последними его версиями меньше проблем).

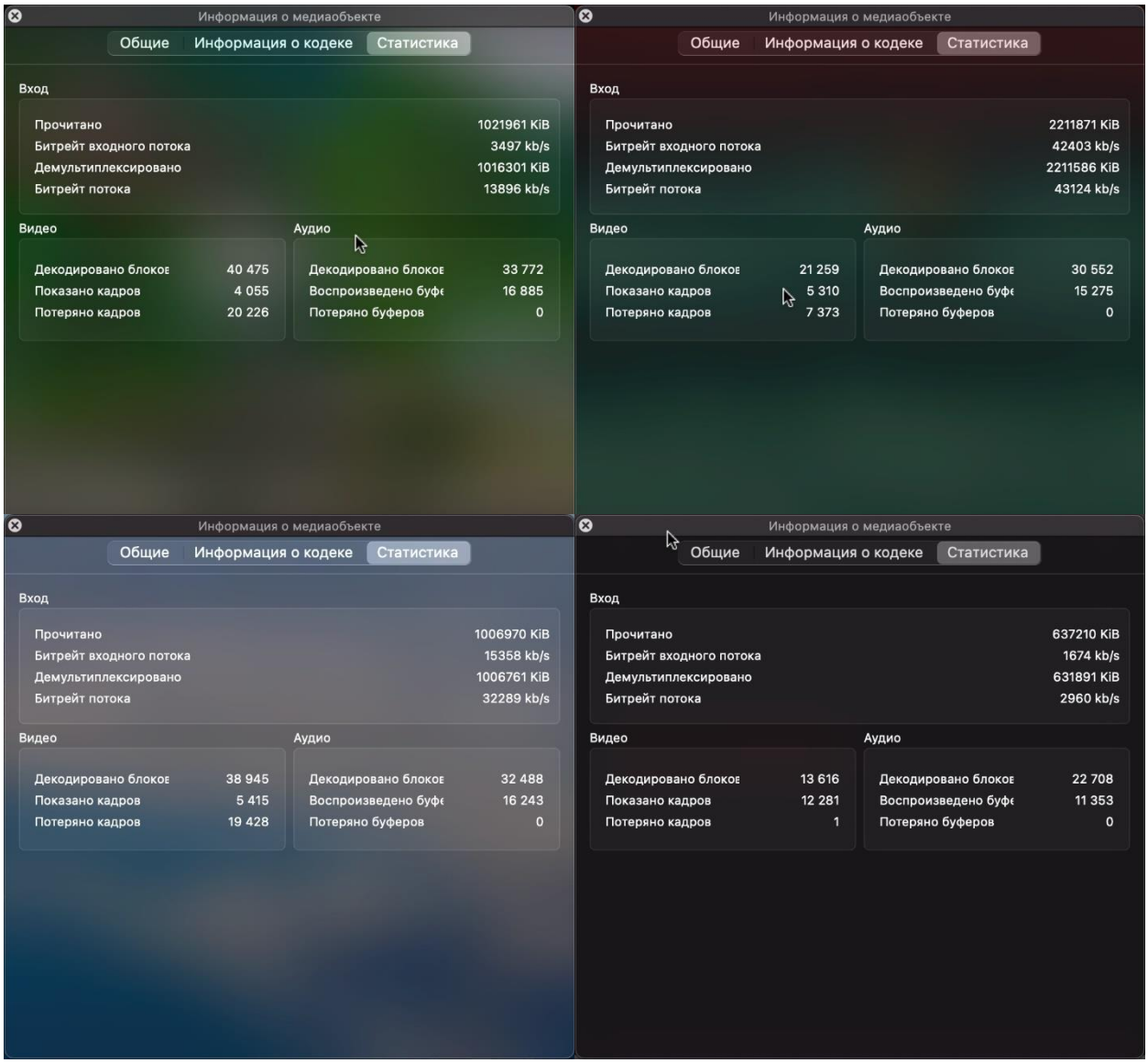
```
unsetopt nomatch; for i in "https://youtu.be/1La4QzGeaaQ"
"https://youtu.be/5LxsViMm25o" "https://youtu.be/zCLOJ9j1k2Y"
"https://youtu.be/2KgaKvpxR0E"; do ./youtube-dl -f bestvideo+bestaudio/best --
ffmpeg-location ./ffmpeg -o ~/Desktop/Transcoding/%(upload_date)s-%(height)sp\ --
-\ %(uploader)s\ ---\ %(title)s.%(ext)s $i; done
```

В начале команда unsetopt nomatch требуется, т.к. без неё zsh (оболочка командной строки) выдаёт ошибку «no matches found».



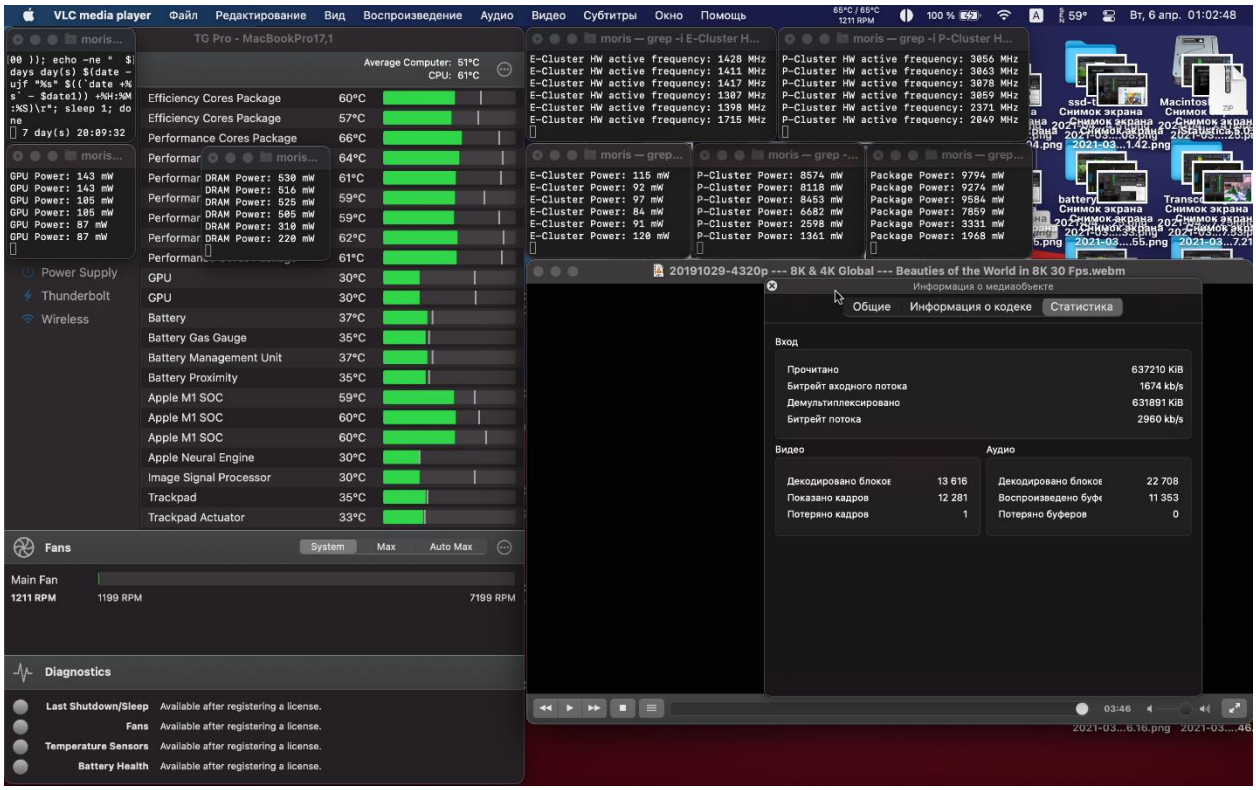
Скриншот 170. Загрузка всех 8K видеороликов из главы 7.4 при помощи youtube-dl.

После того, как мы загрузили себе все видео, порядок действий простой: запускаем их в VLC и открываем статистику.



Скриншот 171. 8K видео в VLC. Слева направо: AV1 HDR 8K60, AV1 HDR 8K30, AV1 SDR 8K60, VP9 SDR 8K30.

Я попробовал воспроизвести эти видео в VLC, и все они, кроме VP9 SDR 8K 30 FPS воспроизводились с большими тормозами. Ну, как с большими, на 2-ух 60 FPS видео я наблюдал просто статичные кадры всё время их проигрывания. С тормозами с горем пополам воспроизвёлся AV1 HDR 8K 30 FPS сэмпл, и на этом всё.

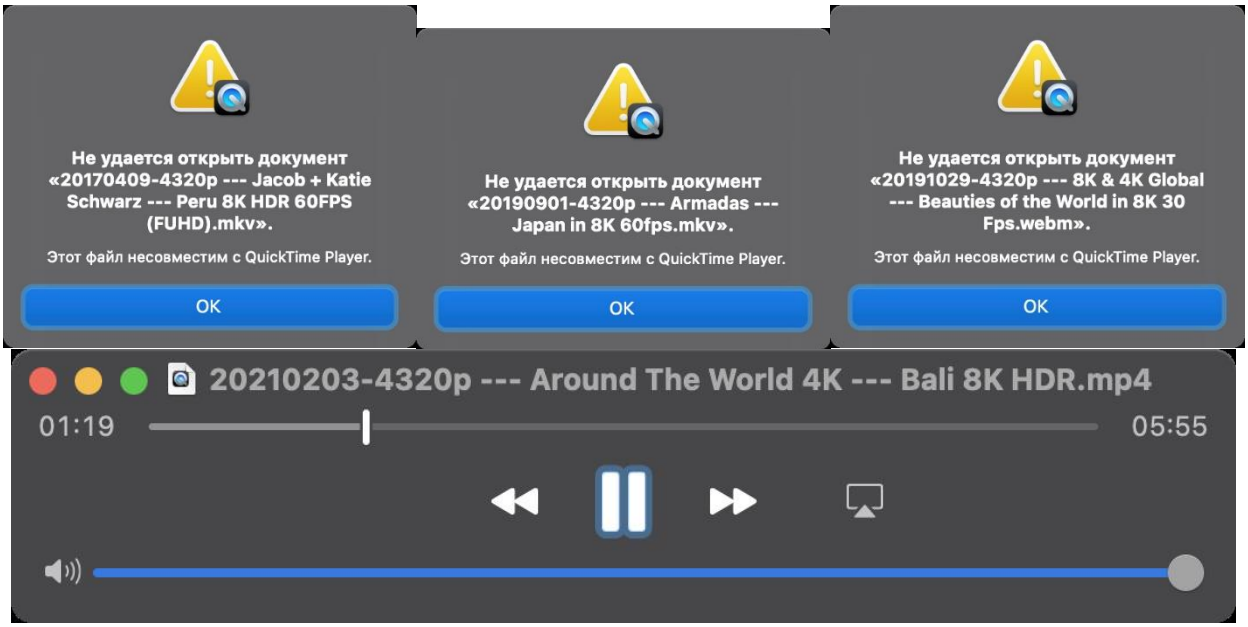


Скриншот 172. Потребление энергии при проигрывании 8K VP9 SDR 30 FPS видео в VLC.

Занятное наблюдение: VLC, как и Chrome, видео с YouTube в кодеке VP9 с разрешением 8K, частотой кадров 30 кадр/сек и глубиной цвета 8-бит (обычный SDR, не HDR) также воспроизводил, судя по всему, задействуя не аппаратные блоки декодирования видео, а центральные ядра CPU.

Т.е. на macOS только Safari может воспроизводить 8K VP9 30 FPS SDR видео, задействуя аппаратные блоки декодирования видео в Apple M1.

А ведь есть же QuickTime? Его ведь Apple сами делают, как и Safari.



Скриншот 173. Попытка воспроизведения 8K контента в QuickTime.

QuickTime отказался воспроизводить 3 сэмпла из 4, а в 4-ом он воспроизвёл аудио, но не видео. В общем, про 8K можно забыть во всех сценариях, кроме воспроизведения 8K SDR 30 FPS контента в кодеке VP9. HEVC (H.265) я уже тестировал и ничего хорошего из этого не вышло. Если для Вас не важно потребление энергии, можете 8K30 видео в SDR смотреть и в Chrome, и в VLC при условии, что оно будет в кодеке VP9. Да, оно будет декодироваться центральным процессором, а не отдельным вычислительным блоком, специально заточенным под эту задачу, но пропуск кадров там минимален, так что, если наплевать на потребление энергии и загрузку процессора, вполне себе вариант.

Эти тесты мы проводили только интереса ради, чтобы понять, имеют ли техническую возможность новые Macbook Pro воспроизводить видео в 8K и насколько хорошо они справляются с этой задачей, способны ли они решать её, задействуя отдельные аппаратные блоки в SoC для ускорения декодирования и перекодирования такого видео, или же всё ограничивается возможностями центрального процессора. Если вы спросите меня, нужен ли мне 8K, то, разумеется, отвечу что нет, т.к. контент (в основном – аниме) я потребляю в 1080p. Да и максимальное разрешение изображения, выводимого на внешний монитор, у Macbook на M1 составляет 6K (6016x3384). Даже если я запущу воспроизведение видео в 8K и выведу его на внешний 6K монитор, по итогу это самое 8K видео будет сжато до 6K для отображения на этом самом 6K мониторе.

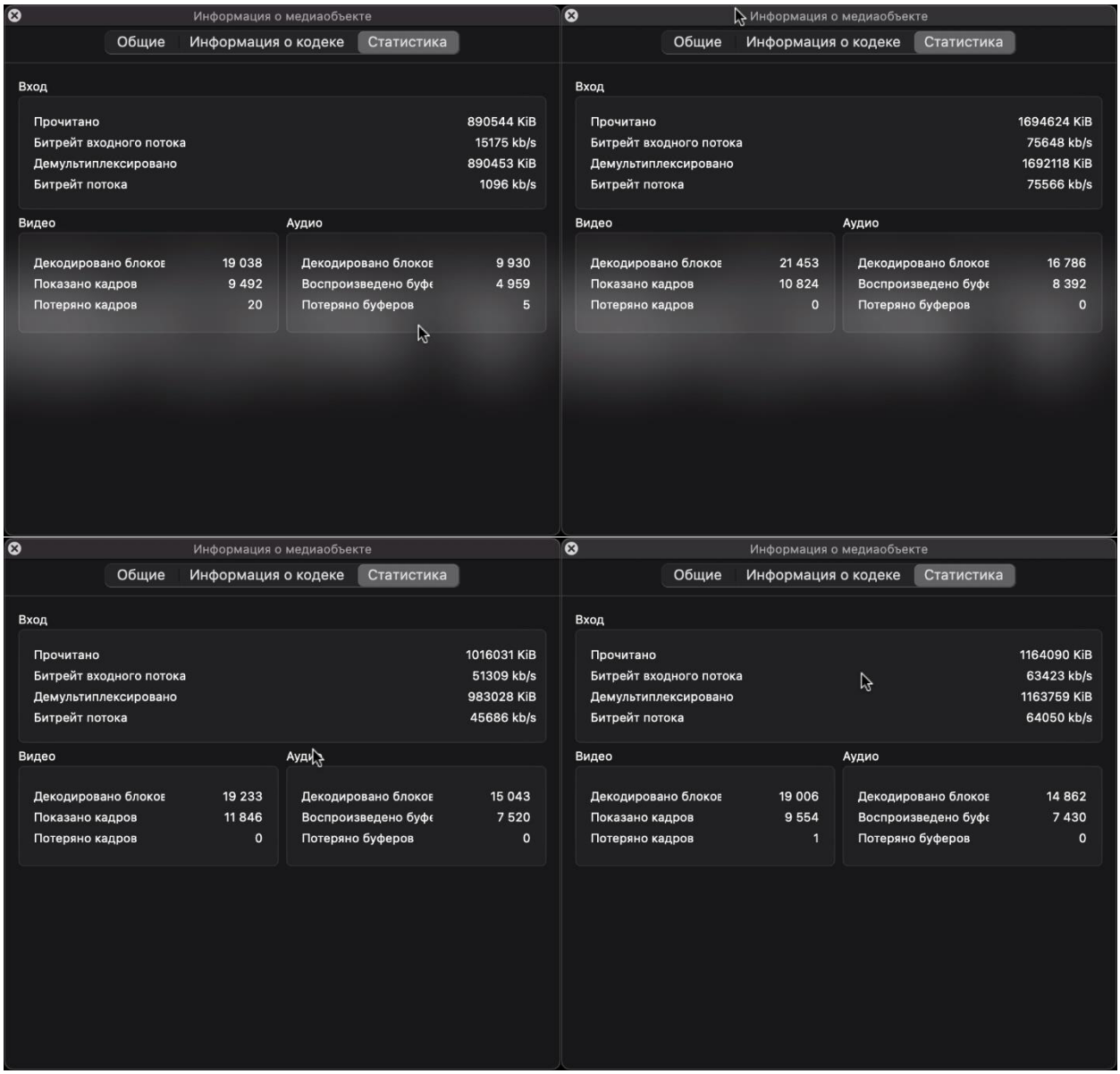
Более чем в 4K я смысла не вижу, и потому мы 4K и протестируем.

На тесте у нас 4 сэмпла. Все сняты в 4K и 60 FPS, но с разными кодеками и разной глубиной цвета (8 бит или 10 бит). Ссылки на них:

1. H.264 (AVC) 8-bit 45,9 Mbit. Прямая ссылка на Google Drive;
2. H.265 (HEVC) 8-bit 77,4 Mbit. Прямая ссылка на Google Drive;
3. H.265 (HEVC) 10-bit 51,6 Mbit. Прямая ссылка на Google Drive;
4. H.265 (HEVC) HDR10 60,1 Mbit. Прямая ссылка на Google Drive.

Сэмпл №2 мы использовали также в тесте ffmpeg.

Запускаем и смотрим на статистику.



Скриншот 174. 4K 60 FPS видео в VLC. Слева направо: H.264 8-bit, H.265 8-bit, H.265 10-bit и H.265 HDR10.

Тут всё вполне предсказуемо. Было бы странно, если бы у ноутбука такого уровня были проблемы с воспроизведением 4K видео в довольно популярных форматах, которые VLC то уж точно умеет аппаратно декодировать. Потерянных кадров 0-20, т.е. терялось не более 0,21% кадров при воспроизведении. Это вполне можно списать на погрешность. У всех ноутбуков, которые мы тестировали ранее в VLC, наблюдались ранее нюансы с тем, что в первые пол секунды воспроизведения 4K 60 FPS H.264 8-bit терялось около десятка или двух десятков кадров, но затем видео воспроизводилось на всех без просадок. Тут та же история.

С 4K никаких проблем нет ни в одном сценарии, так что перейдём к выводам.

7.6. Выводы по декодированию и перекодированию видео.

Опишу ниже выводы, к которым я пришёл, проводя эти тесты.

Да, блоки аппаратного декодирования видео у Apple M1 действительно могут декодировать 8K видео с кодеком VP9 с глубиной цвета 8-бит и частотой кадров 30 FPS. Но могут ли эти блоки аппаратного декодирования обработать видео, сжатое другими кодеками, в том же разрешении 8K? Без понятия. Тут или с софтом несовместимость, т.е. софт не умеет задействовать эти самые блоки аппаратного декодирования видео при разрешении 8K. С такой проблемой мы столкнулись с Chrome, Firefox и VLC: они декодировали видео силами ядер центрального процессора, а не отдельными аппаратными блоками декодирования видео, интегрированными в SoC Apple M1.

Вполне может быть, что поддерживается декодирование и H.265 (HEVC) в 8K 30 FPS, а может и 8K 60 FPS, и аналогично и с кодеком VP9, и с кодеком AV1, но на сегодняшний день нет программного обеспечения, в котором мы могли бы это протестировать, и нет нормальных сэмплов.

В любом случае, при монтаже видео в Final Cut, Davinci Resolve и Adobe Premiere Pro (с применением Adobe Media Encoder) имеется возможность задействовать прокси-медиа. Прокси-медиа — это, по сути, копии оригинальных файлов, но с кодеком, который лучше обрабатывается программой для видеомонтажа (у Final Cut это – ProRes), и, опционально, более низким разрешением. С ProRes Proxy в том же Final Cut, заранее перекодируя видеофайлы можно монтировать 4K ролики даже на Macbook 12 2015-го года, чего уж говорить о Macbook Pro с SoC Apple M1, который опережает его в разы по части центрального процессора и видеоускорителя.

С 4K нет никаких проблем у нового Macbook Pro с Apple M1. Chrome, Firefox (без HDR) и VLC задействуют аппаратные блоки декодирования видео с кодеками H.264 и H.265 в 4K 60 FPS, так что тут всё замечательно.

Единственное, что меня смутило, так это то, что транслированная через Rosetta 2 версия ffmpeg быстрее декодирует видео в H.265, чем нативная, аж до 2 раз быстрее. Удивила и разница между dav1d с Rosetta 2 и без: до 4.5 раз.

В целом же к ноутбуку в этом плане у меня вопросов нет, всё хорошо.

8. Тестирование автономности.

Вот мы и переходим к одной из самых увлекательных частей этого обзора. Преимущество ARM перед x86, как-никак, это намного более низкое энергопотребление. И, разумеется, мы хотим знать, сколь сильно смена архитектуры с x86 на ARM сказалась на автономности Macbook Pro.

Для проведения этого теста я переписал свой PowerShell-скрипт для теста автономности под zsh, являющийся стандартной командной оболочкой в macOS, начиная с версии 10.15. Хотя, так-то можно этот же скрипт запустить и с bash, просто заменив в 1-ой строке zsh на bash. Там отличия минимальные в рамках этого скрипта, я ничего специфичного не использую.

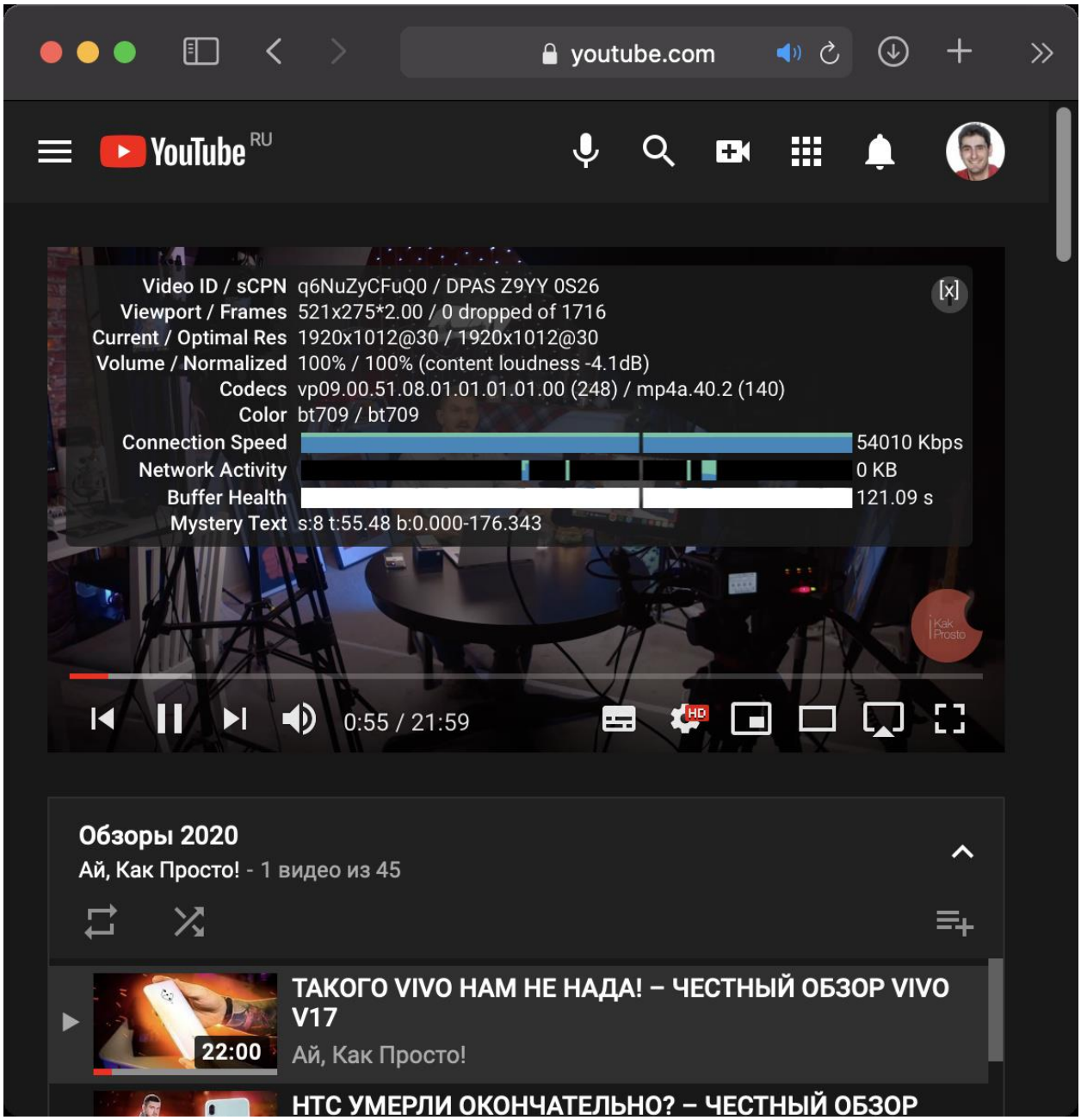
В чём суть zsh-скрипта для теста автономности? Та же, что и у PowerShell-скрипта: мы каждые 10 секунд проверяем, изменился ли заряд батареи, и, если изменился, то пишем в файл “battery_test_log.txt” информацию о времени, когда процент заряд изменился, и, собственно, новый процент заряда. Т.е., например, если в 15:00:00 аккумулятор был заряжен на 100%, а в 15:05:00 аккумулятор разрядился до 99%, то в файл будет записано «99;15:05:00». Далее мы сгружаем эти данные в Excel и просто подсчитываем общие и средние показатели. Всё просто.

Если Вам не нравится идея со скриптом, не вопрос, можете поковырять логи операционной системы и поискать самим, в какой момент ноутбук разрядился и когда он закончил заряжаться. Мне удобнее использовать скрипт и всю нужную мне информацию сгружать в отдельный файл. Скрипт выполняет проверку раз в 10 секунд и пишет в файл информацию только при наличии изменений, его влияние на результат минимально, менее 1%.

Открываем Chrome или Firefox, ставим расширения uBlock Origin и YouTube Auto HD + FPS. Последнее нам потребуется для принудительной установки разрешения 1080p, иначе периодически видео может стартовать в 720p или 4K. Открываем YouTube-канал Ай, Как Просто, запускаем плейлист «Обзоры 2020» и зацикливаем воспроизведение. Вот, собственно, и весь тест. Потом по файлу, сгенерированному скриптом, смотрим, когда у нас начался тест и когда закончился, т.е. когда ноутбук разрядился или зарядился.

Мы провели в Chrome и Firefox по 3 теста со стандартным кодеком AV1, с кодеком H.264 при помощи расширения enhanced-h264ify, которое позволяет блокировать другие кодеки, а также с кодеком VP9 при помощи того же расширения enhanced-h264ify.

В случае с Safari есть нюанс.

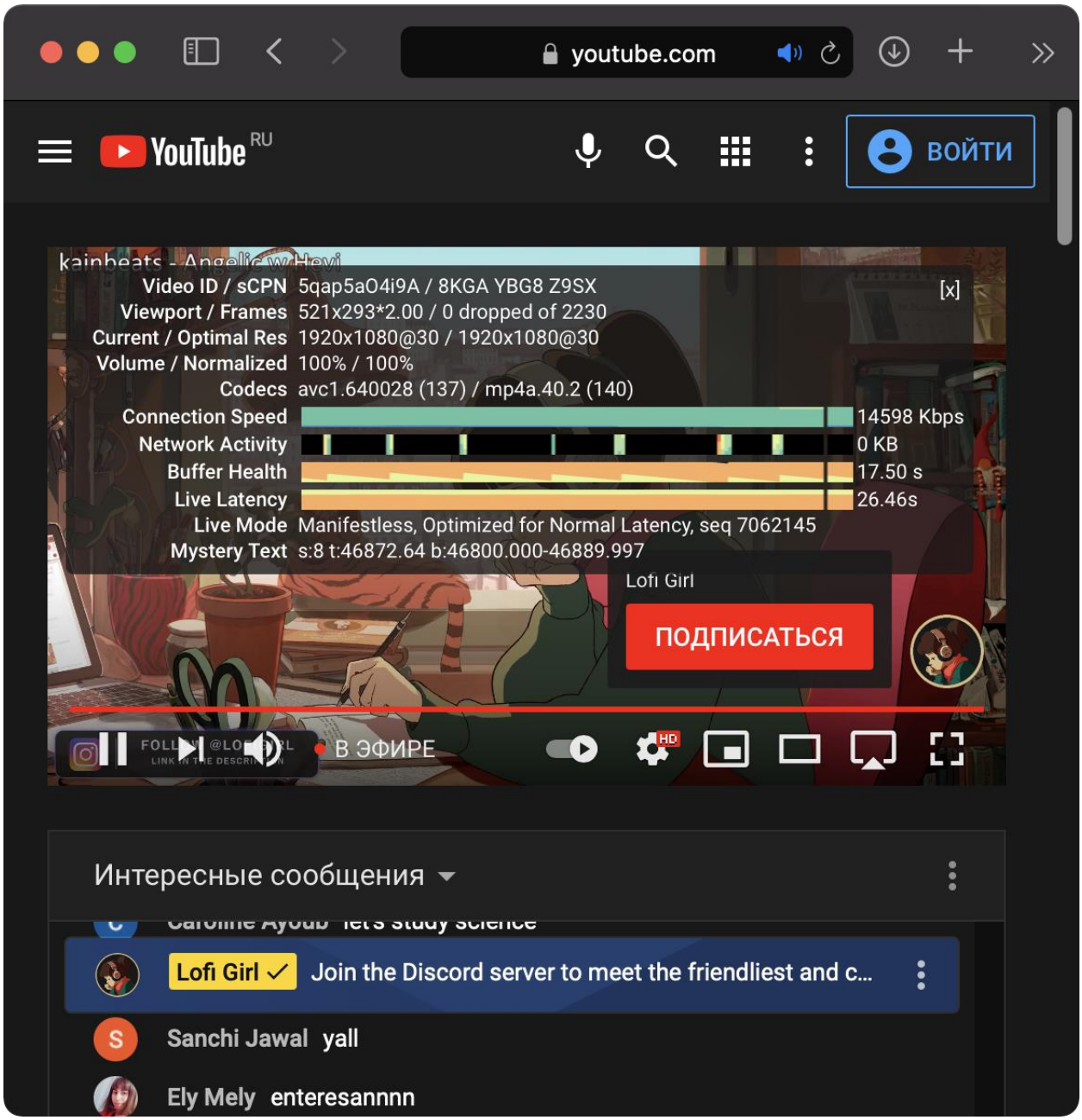


Скриншот 175. Safari. Кодек по умолчанию при воспроизведении видео в 1080p.

Если Вы читали главу 7.4, то знаете, что у Safari отсутствует поддержка воспроизведения видео в кодеке AV1, а YouTube в 1080p умеет транслировать видео всего в 3 кодеках: AV1, VP9 и H.264. Чтобы определить, какой же кодек используется при воспроизведении видео с YouTube в Safari, достаточно нажать ПКМ на видео и открыть статистику для сисадминов.

В стоке Codecs Вы можете заметить “vp09”, что говорит нам как-раз о том, что воспроизводится видео в кодеке VP9.

А какой кодек используется при проигрывании стримов с YouTube?

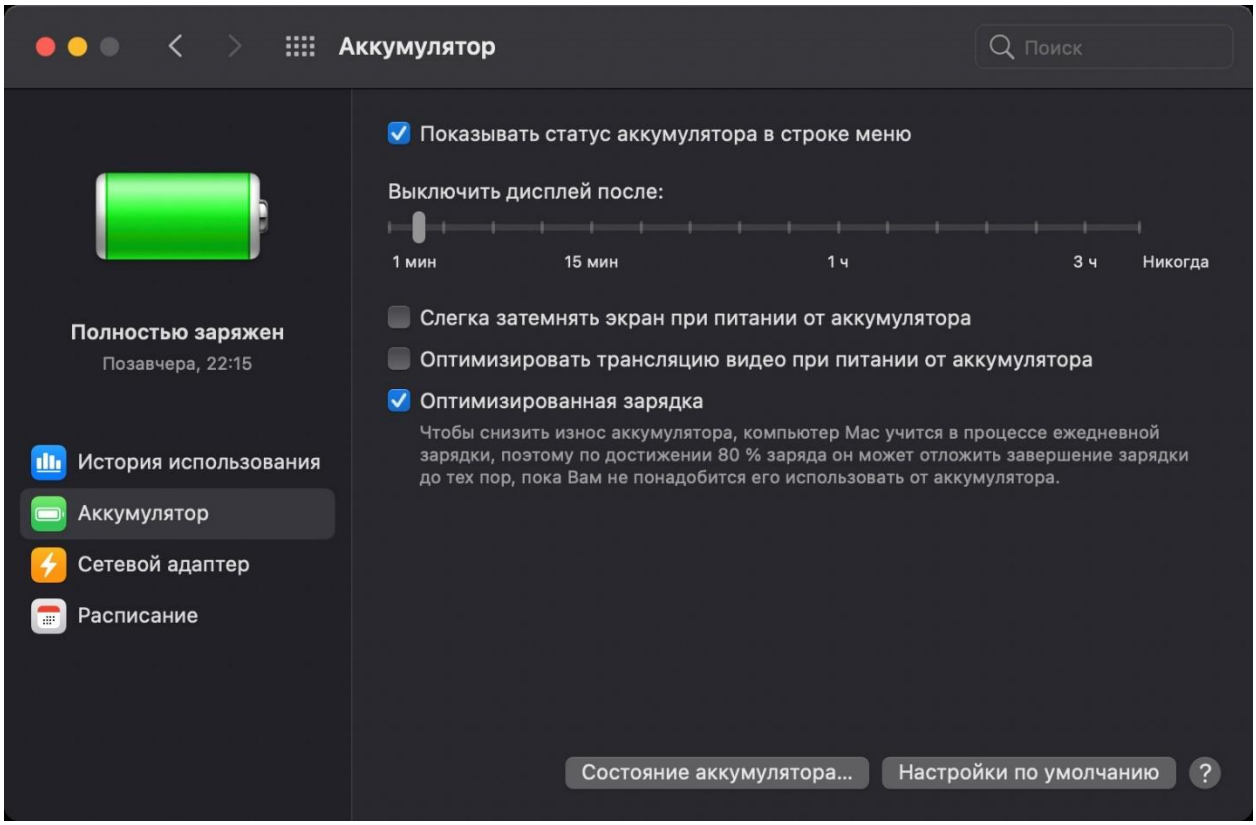


Скриншот 176. Safari. Кодек по умолчанию при воспроизведении стрима в 1080p.

С стримами немного иначе: тут задействуется уже кодек H.264 (он же AVC, он же avc1 в строке Codecs). Для обоих указанных кодеков имеются блоки аппаратного ускорения видео не только у Apple M1, но и у всех x86 и даже распространённых ARM SoC для смартфонов и планшетов последних 3 лет. В ноутбуках декодирование H.264 было аппаратно реализовано ещё в моделях первого поколения Intel Core с интегрированным видеоускорителем.

Разумеется, с H.264 кодеком в 1080p ни у кого не может быть проблем.

Следующий нюанс в случае с Safari: под него нет расширения YouTube Auto HD + FPS и я не нашёл никаких аналогов этого расширения. Это значит, что в случае с Safari мы не можем быть уверены в том, что видео всё время воспроизводилось в разрешении 1080p и временами браузер не переключал разрешение воспроизводимого видео на 720p или ниже. Т.е. у нас при тестировании Safari существует такой фактор неопределённости.



Скриншот 177. Пункт "оптимизировать трансляцию видео при питании от аккумулятора" не активен в настройках.

Тест мы проводим на уровне яркости в 56%, что примерно соответствует яркости в 200 кд/м2. Громкость динамиков – 19%. Пункт «оптимизировать трансляцию видео при питании от аккумулятора» я не стал задействовать в тесте Safari, т.к. нам и без того хватает неопределённости с разрешением воспроизводимого видео, и нет чёткой гарантии, что видео время от времени не будет воспроизводиться с кодеком H.264 вместо VP9. Браузер ведь поддерживает декодирование обоих кодеков, и YouTube в 1080p вещает в обоих этих кодеках, так что однозначно указать кодек мы не можем.

Я 1 раз активировал пункт «оптимизировать трансляцию видео при питании от аккумулятора», и результат в Chrome изменился менее чем на 1%.

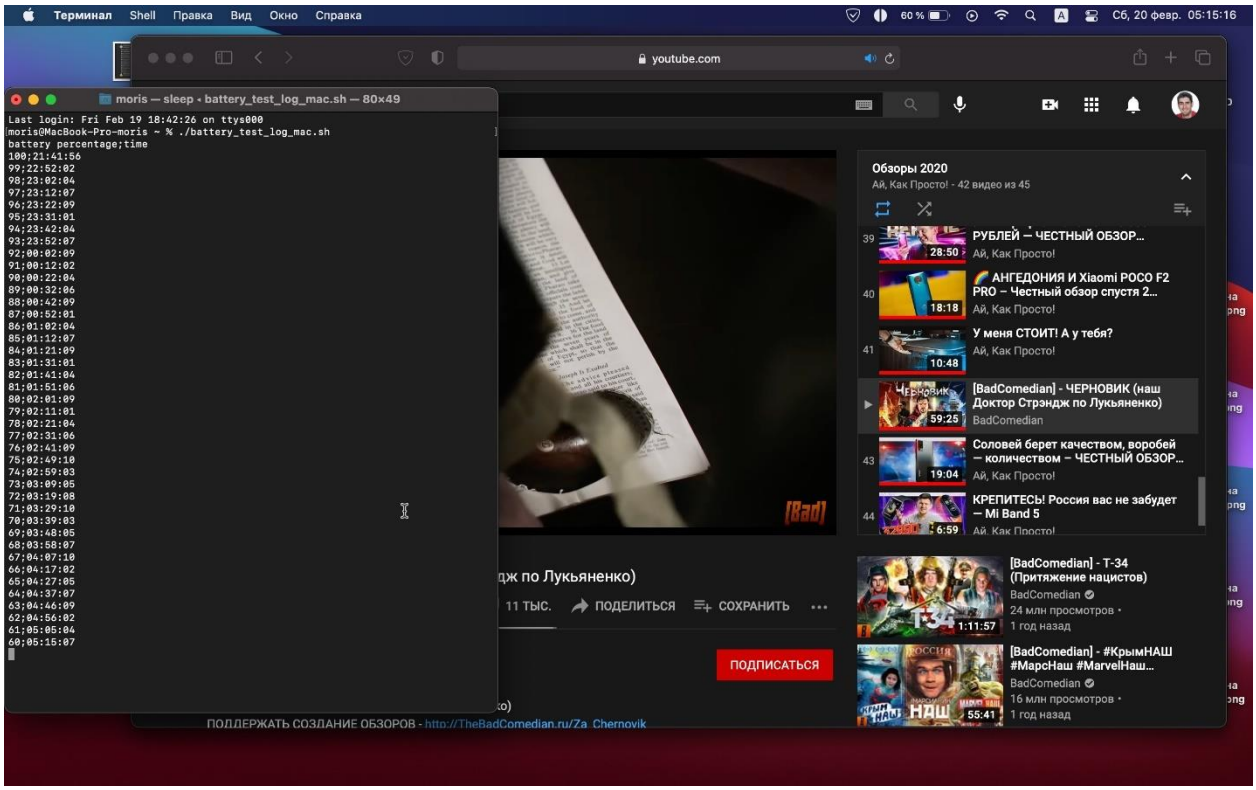
Итак, что по итогу? Сведём все результаты в сравнительную таблицу:

Laptop	Scenario	AC/DC	Battery capacity (W*h)	Avg rate (W)	Avg rate (%/min)	Avg period (min/%)	Total time
Asus ROG Strix Scar 17 G732LXS-HG059T	YouTube video in Chrome	Discharge	66	18,22	0,46	0:02:10	3:37:21
		Charge		40,45	1,02	0:00:58	1:37:54
	YouTube H.264 video in Firefox	Discharge		21,40	0,54	0:01:51	3:05:02
		Charge		45,27	1,14	0:00:52	1:27:28
	YouTube video in Firefox	Discharge		19,26	0,49	0:02:03	3:25:34
Asus Zenbook Flip UX363EA	YouTube H.264 video in Chrome	Discharge	67,3	9,94	0,25	0:04:03	6:46:06
		Charge		29,67	0,73	0:01:21	2:16:05
	YouTube video in Chrome	Discharge		11,26	0,28	0:03:35	5:58:38
		Charge		29,44	0,73	0:01:22	2:17:11
	YouTube H.264 video in Firefox	Discharge		11,59	0,29	0:03:29	5:48:29
		Charge		29,51	0,73	0:01:22	2:16:50
	YouTube video in Firefox	Discharge		12,64	0,31	0:03:11	5:19:33
		Charge		28,98	0,72	0:01:23	2:19:20
HP Omen 15 (15-en0037u)	YouTube AV1 video in Chrome	Discharge	70,2	9,22	0,22	0:04:34	7:36:41
		Charge		35,65	0,85	0:01:10	1:58:09
	YouTube H.264 video in Chrome	Discharge		6,68	0,16	0:06:18	10:30:52
		Charge		35,50	0,84	0:01:11	1:58:38
	YouTube VP9 video in Chrome	Discharge		7,13	0,17	0:05:54	9:51:08
		Charge		35,19	0,84	0:01:11	1:59:41
	YouTube AV1 video in Firefox	Discharge		9,51	0,23	0:04:25	7:22:47
		Charge		36,72	0,87	0:01:08	1:54:43
	YouTube H.264 video in Firefox	Discharge		8,72	0,21	0:04:49	8:03:17
		Charge		35,89	0,85	0:01:10	1:57:21
	YouTube VP9 video in Firefox	Discharge		8,75	0,21	0:04:48	8:01:11
		Charge		35,75	0,85	0:01:10	1:57:50
Asus ROG Strix G15	YouTube AV1 video in Chrome	Discharge	91,0	10,66	0,20	0:05:07	8:32:05
		Charge		50,21	0,92	0:01:05	1:48:44
	YouTube H.264 video in Chrome	Discharge		9,89	0,18	0:05:31	9:12:16
		Charge		50,24	0,92	0:01:05	1:48:41
	YouTube VP9 video in Chrome	Discharge		9,83	0,18	0:05:33	9:15:37
		Charge		49,96	0,92	0:01:05	1:49:17
	YouTube AV1 video in Firefox	Discharge		11,94	0,22	0:04:34	7:37:19
		Charge		47,59	0,87	0:01:08	1:54:44
	YouTube H.264 video in Firefox	Discharge		11,09	0,20	0:04:55	8:12:10
		Charge		50,38	0,92	0:01:05	1:48:23
	YouTube VP9 video in Firefox	Discharge		11,11	0,20	0:04:54	8:11:32
		Charge		49,83	0,91	0:01:05	1:49:34
Lenovo Yoga Slim 7	YouTube AV1 video in Chrome	Discharge	60,0	5,24	0,15	0:06:51	11:26:30
		Charge		30,46	0,85	0:01:10	1:58:12
	YouTube H.264 video in Chrome	Discharge		4,07	0,11	0:08:50	14:44:45
		Charge		31,01	0,86	0:01:09	1:56:05
	YouTube VP9 video in Chrome	Discharge		4,12	0,11	0:08:44	14:34:26
		Charge		30,81	0,86	0:01:10	1:56:50
	YouTube AV1 video in Firefox	Discharge		6,81	0,19	0:05:17	8:48:42
		Charge		30,13	0,84	0:01:11	1:59:29
	YouTube H.264 video in Firefox	Discharge		6,49	0,18	0:05:32	9:14:59
		Charge		30,42	0,85	0:01:11	1:58:20
	YouTube VP9 video in Firefox	Discharge		6,25	0,17	0:05:45	9:36:15
		Charge		30,24	0,84	0:01:11	1:59:04

Apple Macbook Pro 13 2020 (M1)	YouTube video in Safari	Discharge	58,2	3,47	0,10	0:10:03	16:45:04
		Charge		23,50	0,67	0:01:29	2:28:36
	YouTube video in Safari (Adguard)	Discharge		3,47	0,10	0:10:04	16:47:07
		Charge		23,55	0,67	0:01:28	2:28:16
	YouTube AV1 video in Chrome	Discharge		3,94	0,11	0:08:52	14:47:08
		Charge		23,97	0,69	0:01:27	2:25:42
	YouTube H.264 video in Chrome	Discharge		3,65	0,10	0:09:33	15:56:08
		Charge		23,47	0,67	0:01:29	2:28:46
	YouTube VP9 video in Chrome	Discharge		3,58	0,10	0:09:45	16:16:04
		Charge		23,50	0,67	0:01:29	2:28:37
	YouTube AV1 video in Firefox	Discharge		4,12	0,12	0:08:28	14:07:33
		Charge		24,22	0,69	0:01:26	2:24:11
	YouTube H.264 video in Firefox	Discharge		3,91	0,11	0:08:55	14:52:06
		Charge		23,66	0,68	0:01:28	2:27:36
	YouTube VP9 video in Firefox	Discharge		3,79	0,11	0:09:13	15:22:02
		Charge		23,93	0,69	0:01:27	2:25:56

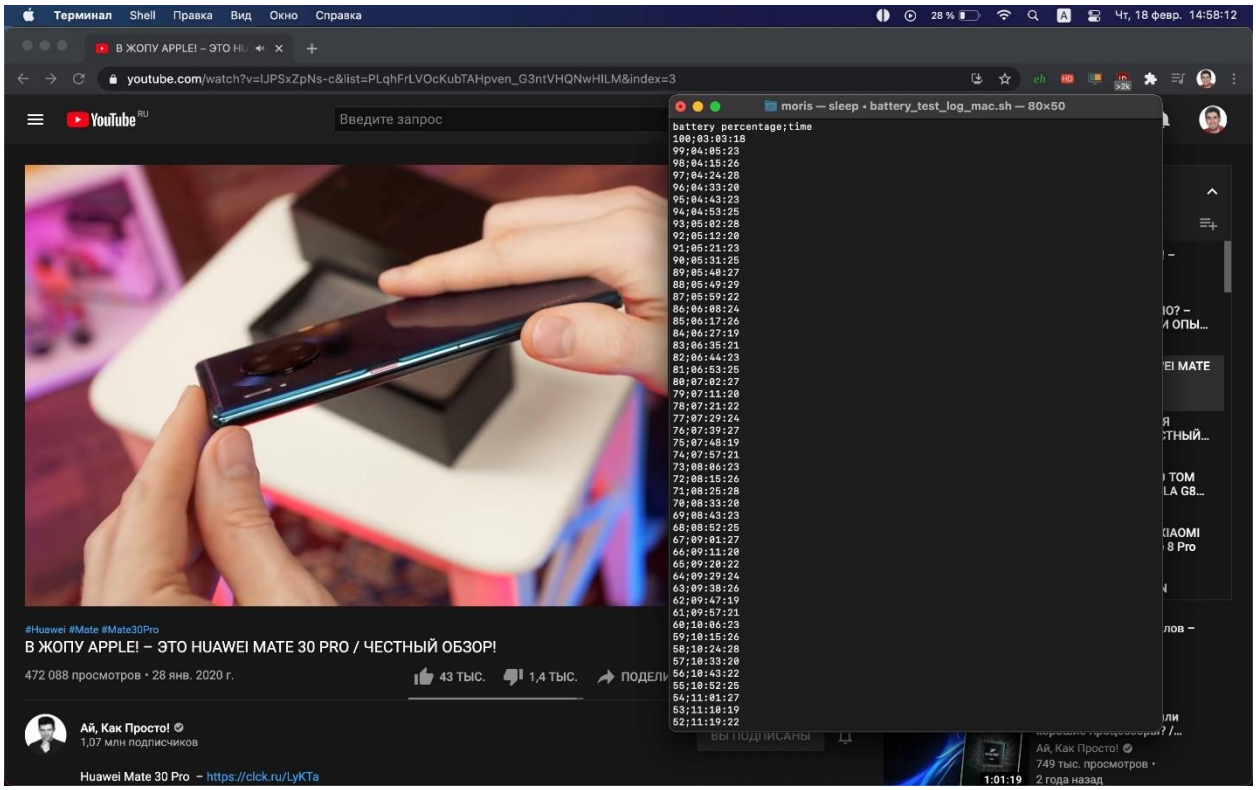
Таблица 5. Сравнительная таблица с результатами тестов автономности ноутбуков.

Что? Как? Куда? Родите меня обратно. 16 часов при воспроизведении 1080p видео с YouTube в Chrome и 16.75 часа при воспроизведении YouTube видео в Safari (за разрешение в Safari ручаться не могу). Это №1 результат в наших тестах. Получается, что в течение 16 часов тестов весь ноутбук, включая экран, модуль сигнала Wi-Fi, SoC, включая и оперативная память, SSD и другие компоненты на материнской плате совокупно потребляли около 3.5 Ватт. Как Вы этого добились, Apple? Предыдущим рекордсменом у нас был Lenovo Yoga Slim 7 со средним потреблением 4.1 Ватт, а у Mac – 3.5.



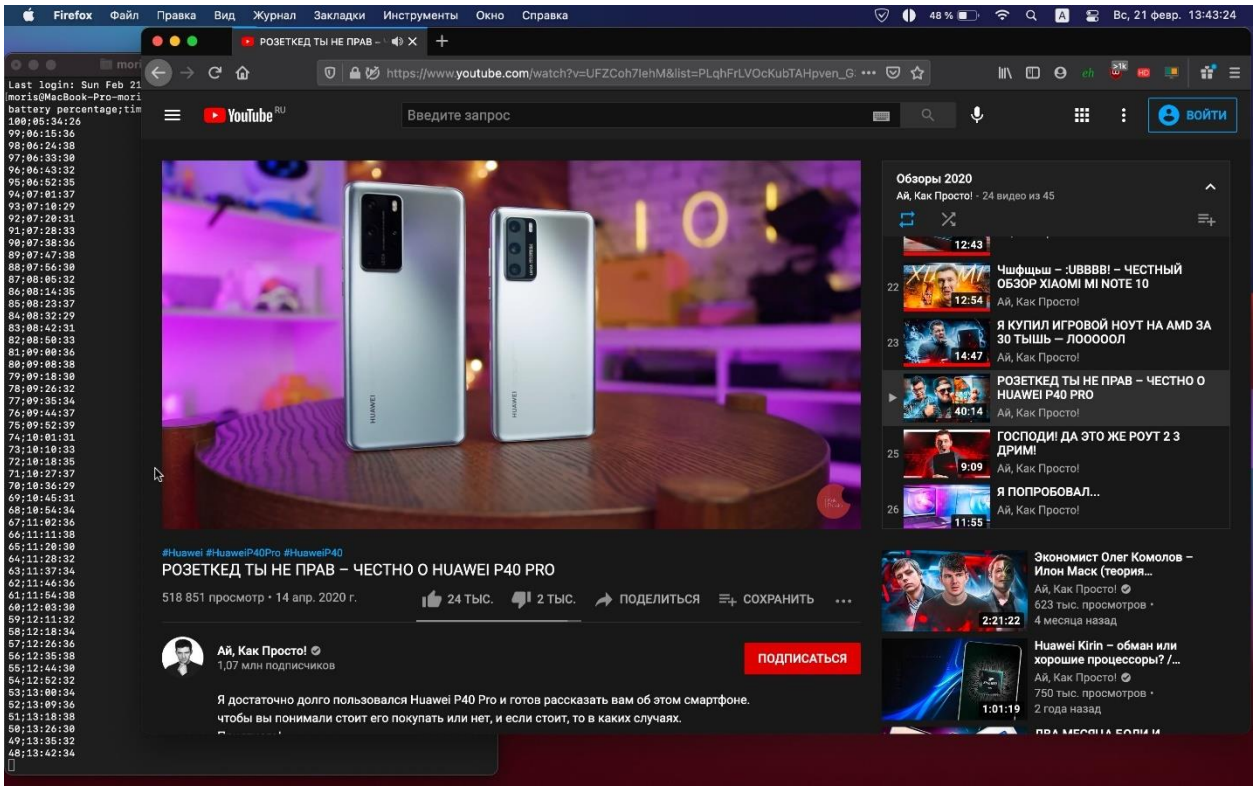
Скриншот 178. Тест автономности Macbook Pro 13 2020 (Apple M1). Браузер Safari.

Я провёл 2 теста в Safari: с блокировщиком рекламы Adguard и без него: разница составила всего 2 минуты, что легко списывается на погрешность.



Скриншот 179. Тест автономности Macbook Pro 13 2020 (Apple M1). Браузер Chrome. Кодек H.264.

Занятно, что 1-ый 1% заряда аккумулятора новый Macbook Pro с Apple M1 теряет за 40-60 минут, т.е. проходит пол часа или час с начала работы от аккумулятора прежде, чем заряда опускается со 100% до 99%. А потом уже 1% заряда теряется за примерно за 9-10 минут. В этом нет ничего плохого, что заряд со 100% до 99% расходуется столь медленно, просто это стоит иметь в виду и не думать «а, у меня до сих пор 100%, значит, разрядится он только часов через 40». Нет, просто конкретно первый % теряется так долго. Если бы мы, как некоторые другие тестировщики, проводили тест автономности, делая выводы лишь за час тестирования (мол, вот за час проигрывания видео в YouTube, ноутбук разрядился на 1%, значит, его хватит на 100 часов), думаю, Вы и сами понимаете, какими абсурдными были бы выводы. Поэтому мы и тестируем гаджеты, разряжая их со 100% до значения, близкого к 0%. Строго до 0% мы почти никогда не разряжаем аппараты, т.к. чаще всего ноутбуки на Windows уходят в гибернацию при 5%, а вот с Mac – иначе, ведь он работал ещё минут 10 даже при заряде в 1%.



Скриншот 180. Тест автономности Macbook Pro 13 2020 (Apple M1). Браузер Firefox. Кодек H.264.

С воспроизведением видео Firefox справился не столь эффективно, его хватило не на 16.75 часов и не на 16 часов, а на 15 часов. Впрочем, это, всё равно, отличный показатель. В этом ничего удивительного также нет, т.к. и предыдущие ноутбуки на нашем тесте с Firefox жили меньше, чем с Chrome.

После этого я решил, что маловато нагрузки дал на Macbook с M1, и провёл тесты автономности с воспроизведением 4K видео чисто из интереса ради: насколько хватит аккумулятора, если значительно повысить нагрузку?

Laptop	Scenario	AC/DC	Battery capacity (W*h)	Avg rate (W)	Avg rate (%/min)	Avg period (min/%)	Total time
Apple Macbook Pro 13 2020 (M1)	YouTube 4K AV1 video in Chrome	Discharge	58,2	5,21	0,15	0:06:42	11:10:05
		Charge		23,05	0,66	0:01:30	2:31:29
	YouTube 4K VP9 video in Chrome	Discharge		5,62	0,16	0:06:12	10:21:06
		Charge		24,15	0,69	0:01:26	2:24:34
	YouTube 4K AV1 video in Firefox	Discharge		5,38	0,15	0:06:29	10:49:08
		Charge		24,17	0,69	0:01:26	2:24:28
	YouTube 4K VP9 video in Firefox	Discharge		4,17	0,12	0:08:22	13:57:09
		Charge		24,16	0,69	0:01:26	2:24:33

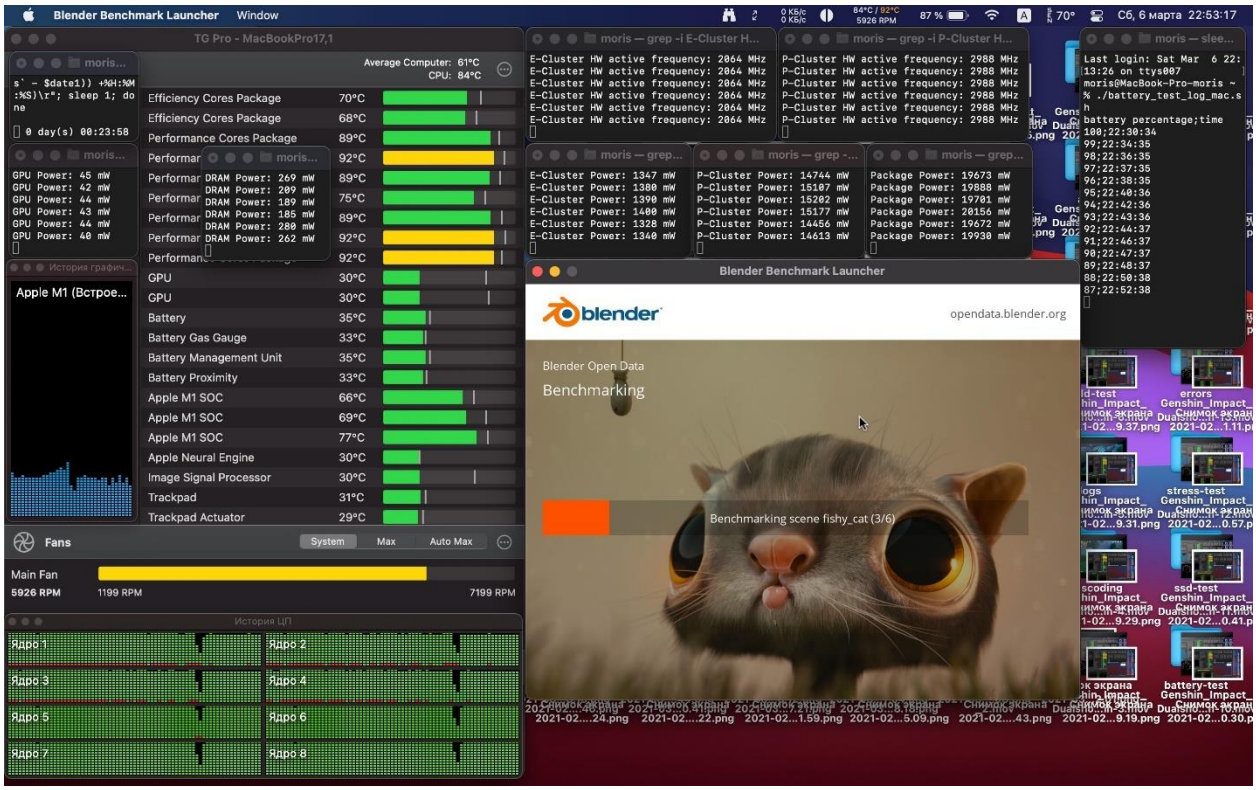
Таблица 6. Тест автономности Macbook Pro 13 2020 (Apple M1) при проигрывании 4K видео в Chrome и Safari.

В случае с проигрыванием видео в 4K, напротив, при воспроизведении в кодеке VP9, который оба браузера способны декодировать выделенными аппаратными блоками, Firefox отработал на целых 3 часа больше: 14 часов.

Да, ноутбук с Firefox, как и в случае с Chrome, при воспроизведении 4K видео прожил меньше, чем при воспроизведении 1080p видео, но с Firefox при воспроизведении видео в кодеке VP9 он прожил всего на 1.5 часа меньше, а с Chrome – на 6 часов меньше.

К слову, я не совсем понял, как так получилось, но с кодеком VP9, который Chrome умеет аппаратно декодировать, в разрешении 4K тот потреблял больше энергии, чем с кодеком AV1, который он декодировать аппаратно не способен. Возможно, это поправят с обновлениями, ведь по идее разница между 4K и 1080p у него должна была быть, как и у Firefox.

В любом случае, если берёте с собой Macbook на мероприятие и Вам требуется на этом мероприятии подключать ноутбук к проектору или к монитору для вывода на него видео в 4K, и зарядки у Вас с собой нет, на данный момент, для Вас Firefox будет более оптимальным решением.



Скриншот 181. Тестирование автономности под максимальной нагрузкой. Blender Benchmark 2.04 (Blender 2.92).

Далее я проверил и то, сколько Macbook проживёт при циклическом повторении теста в Blender Benchmark. Допустим, Вы всю ночь работали и приготовили проект, Вам осталось его ещё отрендерить, но Вы уже не успеваете, Вам пора собираться и выезжать. Насколько хватит его аккумулятора в дороге под максимальной нагрузкой?

Сразу скажу, что закрывать работающий ноутбук и бросать к себе в портфель, т.е. в изолированное пространство, я крайне не рекомендую, он у Вас там может перегреться и выйти из строя при частом повторении подобных действий. Да и забор воздуха в ограниченном пространстве будет весьма посредственным, что скажется на и без того не лучшем охлаждении в Macbook Pro с Apple M1. Но иногда бывают ситуации, когда иначе никак. Сам ноутбук не закрывал, оставил с включенным экраном работать под максимальной нагрузкой и посмотрел, на сколько в итоге его хватило.

Laptop	Scenario	AC/DC	Battery capacity (W*h)	Avg rate (W)	Avg rate (%/min)	Avg period (min/%)	Total time
Apple Macbook Pro 13 2020 (M1)	Blender Benchmark (Blender 2.92)	Discharge	58,2	24,23	0,69	0:01:26	2:24:08
		Charge		21,66	0,62	0:01:36	2:41:15

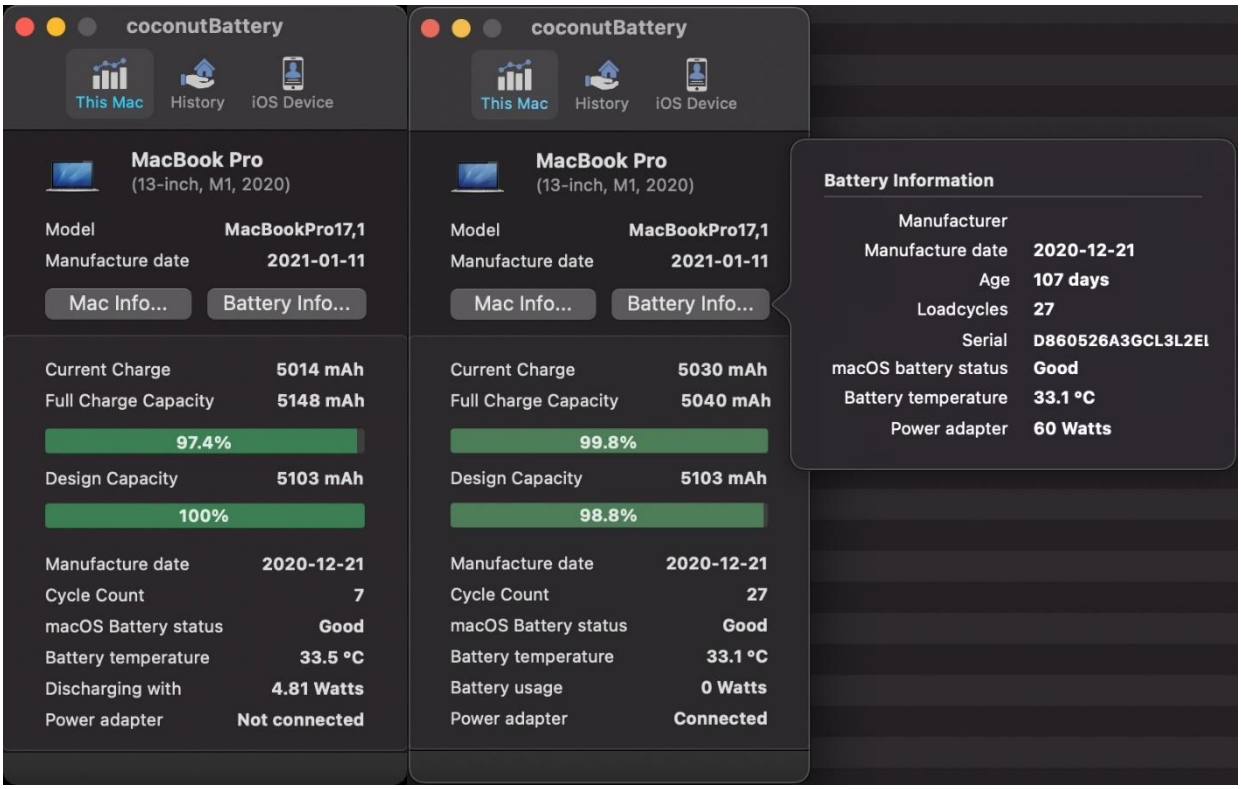
Скриншот 182. Тестирование автономности Macbook Pro 13 (Apple M1) под нагрузкой. Blender 2.90.

У меня он в таком сценарии прожил чуть меньше 2.5 часов. Ну и на зарядку под нагрузкой ушло чуть больше 2.5 часов, тогда как ранее уходило чуть меньше времени. В среднем энергопотребление ноутбука в таком режиме составило 24 Ватта, это с на 100% загруженным CPU и включенным экраном на яркости 56% (около 200 нит)

Тут есть ещё 1 спорный момент: в своём видео на 26:01 Роман Ястреб рассказал про то, что под нагрузкой из-за кулера Macbook Pro разряжается быстрее, чем Air. Безусловно, кулер вносит свою лепту, но Роман забывает про то, что Air под длительной нагрузкой троттлит, а Pro – нет. Если не задействовать термопрокладку между теплораспределительной площадкой в Air и крышкой ноутбука, Air может сбросить за 2 часа более 20% от своей исходной производительности, что, безусловно, негативно скажется на производительности, но позитивно скажется на энергопотреблении, т.к. снижение тепловыделения во избежание перегрева означает и снижение энергопотребления, отсюда и такой результат: 3 часа у Air против 2 у Pro.

Остались 2 вопроса:

1. Насколько аккумулятор износился после тестов автономности?
2. Какова производительность ноутбука при работе от батареи?

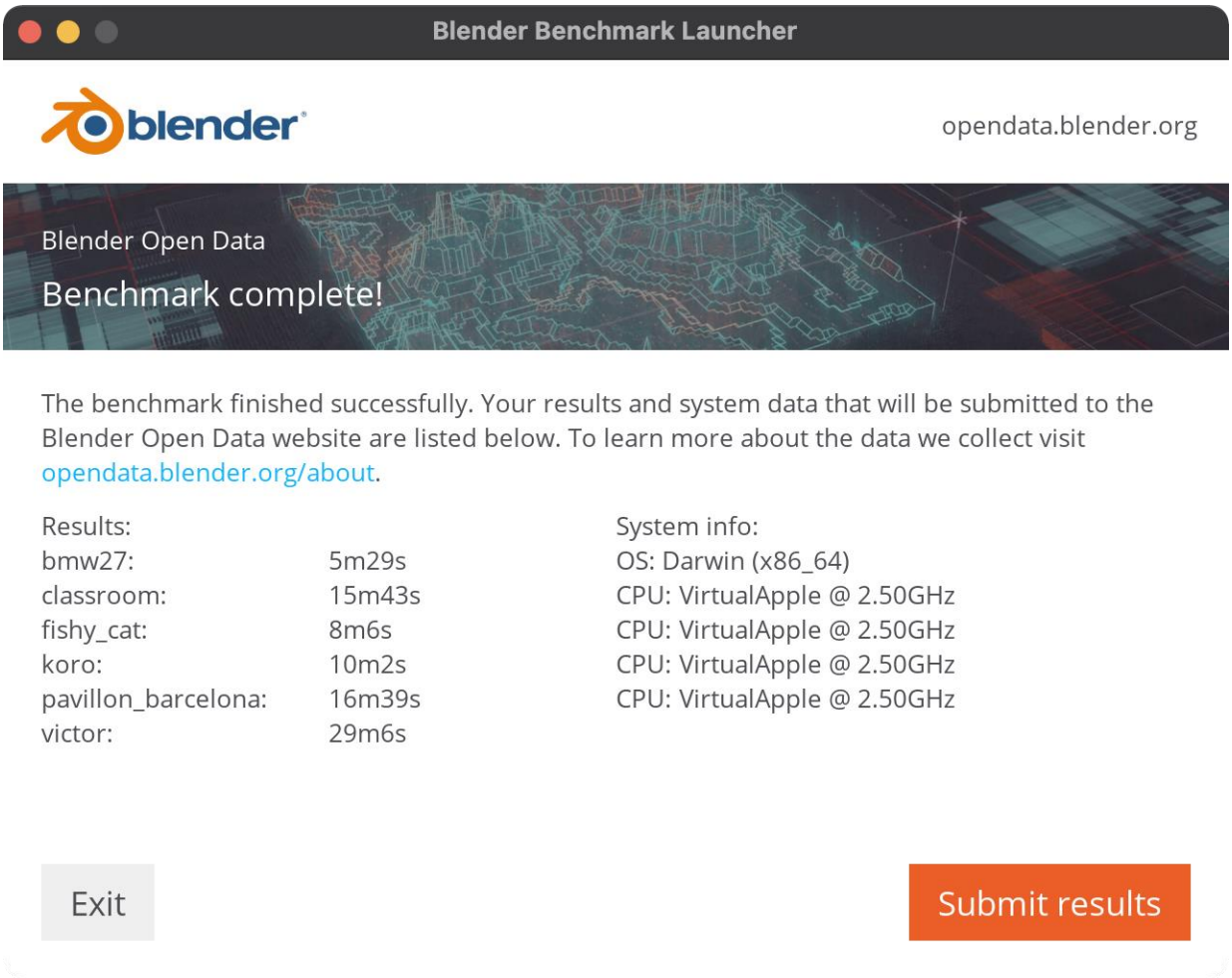


Скриншот 183. Износ аккумулятора до и после тестирования ноутбука.

После 27 циклов перезаряда в ходе нашего тестирования аккумулятор ноутбука потерял всего 2,1% от своей изначальной ёмкости. Он приехал к нам с аккумулятором ёмкостью 100.9% (5148 мА*ч) от номинала, т.е. того значения, которое должно быть с завода (5103 мА*ч), а сейчас его ёмкость составляет 99.8% от номинала (5040 мА*ч).

У меня был на тесте отвратительный ноутбук Dell XPS 17, который потерял 3.7% от своей ёмкости за 5-6 циклов перезаряда, были на тесте и ноутбуки с качественным аккумулятором, такие как HP Omen 15, и Asus ROG Strix G15, у которых износ аккумулятора составил менее 1%, но загвоздка в том, что циклов перезаряда у них было не более 7-8. Ни один аккумулятор в ноутбуках из тех, что я тестировал, не поддавался ещё настолько жёсткому тестированию, как аккумулятор Macbook Pro 13 с SoC Apple M1, и, при этом, его износ составил всего 2,1%. Если сравнивать с тем, какой аккумулятор у него должен быть с завода, он сейчас изношен всего на 1,2%. Это замечательный результат.

Ну и ответим далее на последний вопрос по аккумулятору: насколько ноутбук производителен без наличия питания от электросети?



Скриншот 184. Macbook Pro 13 2020 (Apple M1, 20W). Результат CPU в Blender 2.92 при питании от батареи.

Ранее я приводил то, сколько живёт ноутбук от батареи под максимальной нагрузкой. Мы выбрали тест Blender, т.к. в Blender, т.к. он наилучшим образом имитирует реальную задачу, ведь многие люди работают в Blender и для них скорость рендеринга в нём имеет значение. Мы уже посмотрели на то, сколь долго живёт ноутбук при питании от батареи под такого рода нагрузкой, но с какой производительностью он умудряется 2.5 часа работать от батареи? Итог: разница при питании ноутбука от сети и при питании ноутбука от батареи составила менее 1% в каждой сцене. Я в шоке.

Выводы по автономности: Best of the best, 11/10 на кончиках пальцев Логвинова. А что Вы думали, я скажу о ноутбуке, который живёт 16 часов от аккумулятора при воспроизведении видео в 1080p и целых 2.5 часа при рендеринге в Blender? Мало того, у него и аккумулятор очень медленно изнашивается, так ещё и его производительность нисколько не урезается при питании от аккумулятора. У нас на тесте ещё не было аналогов, равных ему.

9. Какое ПО (не) работает на Mac на Apple Silicon?

9.1. Родное ПО под macOS с Apple Silicon.

Ну, тут всё предельно ясно: ПО под родную платформу, по большей части, пашет без серьёзных проблем. Хотя, тут всё зависит от того, избавился ли разработчик от палок в колёсах в виде изменений в API для M1 от Apple. Да, есть нюансы, к примеру ffmpeg, скомпилированный под ARM для macOS, в главе 7.2 декодирование видео на центральном процессоре проводил медленнее, чем его же версия для x86, прошедшая предварительную трансляцию под ARM на самом ноутбуке через Rosetta 2. Тут всё упирается исключительно в то, насколько хорошо под ARM адаптировано ПО.

Есть также нюансы с тем, что одни и те же API на Mac с Apple Silicon и Intel дают в ответ на запрос разную информацию. Я это описал в главе 2.7.

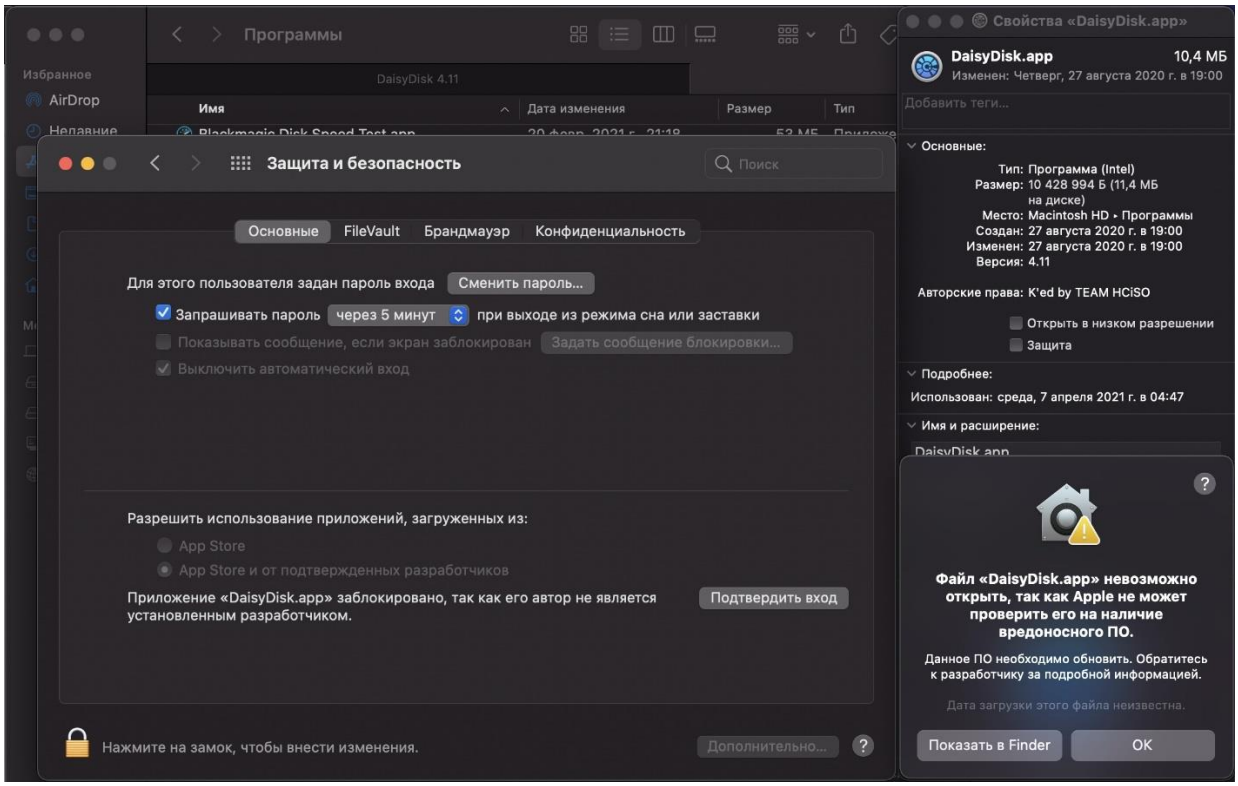
Однако в целом всё должно быть более-менее хорошо.

С чем будут проблемы, так это с кряками. Сам я не проверял, но мой знакомый проверил у себя на Macbook с M1 и поделился со мной отзывом.



Скриншот 185. Проблемы с кряком универсальных приложений. Пример с DaisyDisk.

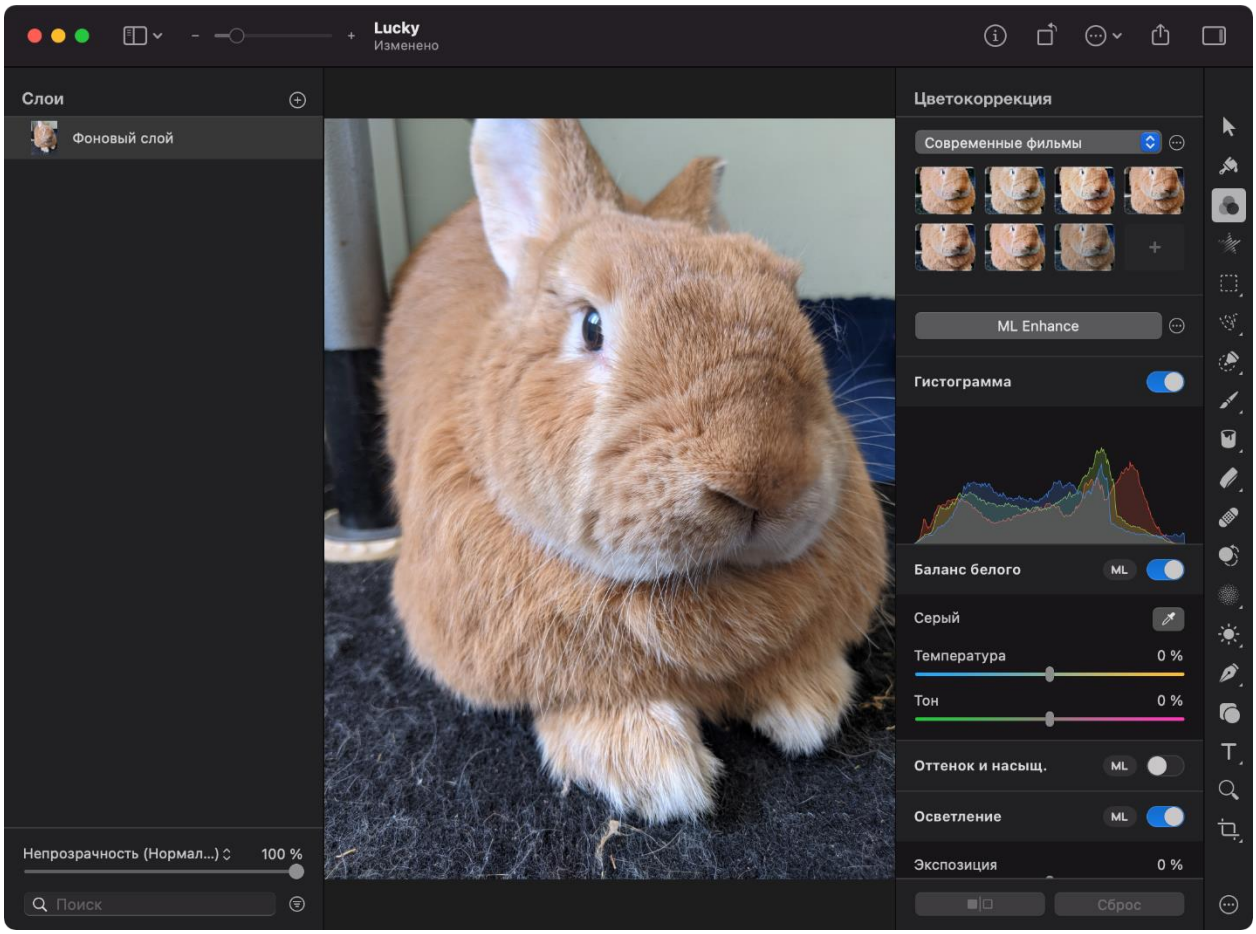
Знакомый привёл пример с утилитой DaisyDisk. Не работают кряки с версией 4.12.1, которая поддерживает macOS Big Sur и которая должна поддерживать нативно Mac с Apple Silicon, т.к., как Вы видите, тип программы – Universal. Причём, не работает как нативно, так и через Rosetta 2. При запуске вылезает сообщение о повреждении программы, решения нет.



Скриншот 186. Запуск старой Intel-версии DaisyDisk.

При этом, с крякнутым DaisyDisk версии 4.11 под Intel (без поддержки M1) проблем нет: знакомый получил предупреждение о возможном вредоносном ПО при запуске, но далее, как он разрешил запуск приложения в настройках macOS, оно заработало через Rosetta 2 без каких-либо проблем.

В чём тут причина, утверждать не берусь. Знакомый поделился информацией, что часть крякнутых приложений работает как положено, а часть – нет, и не работает даже через Rosetta 2.



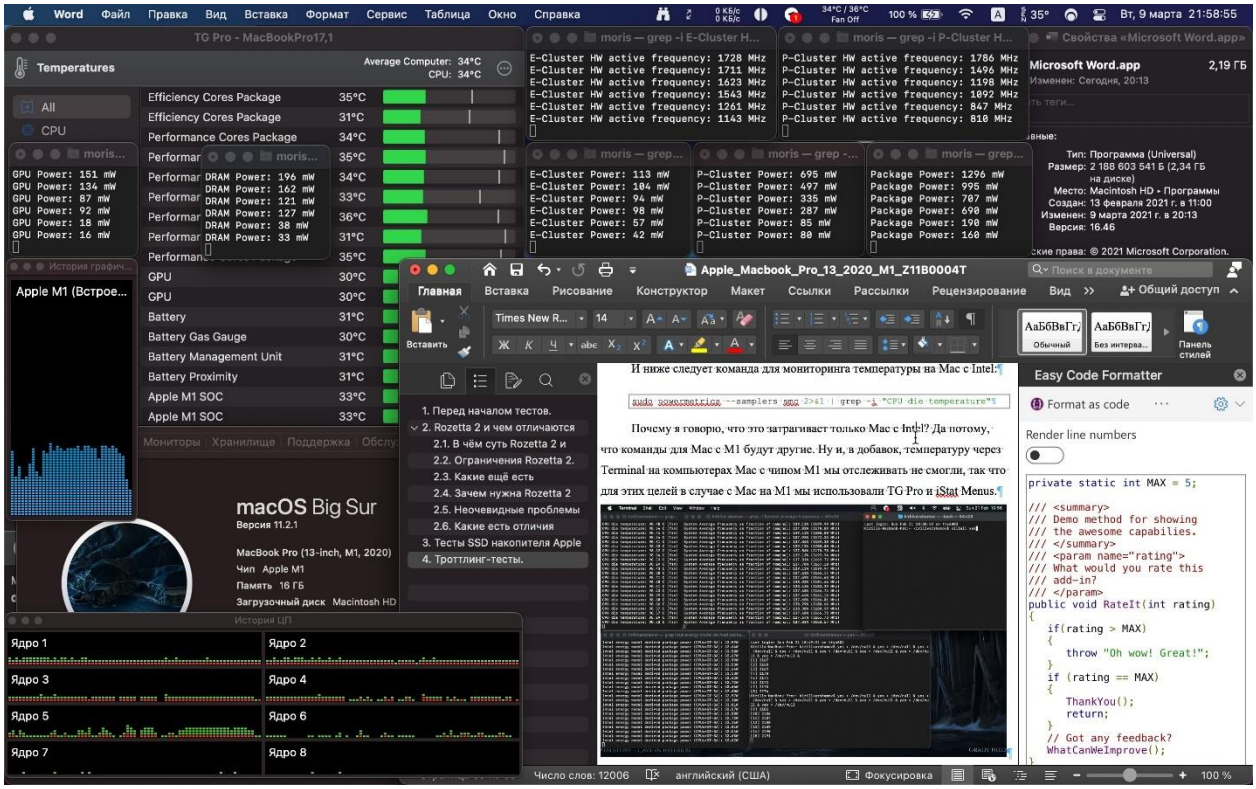
Скриншот 187. Обработка фото моего кролика в Pixelmator Pro 2.0.4.

Как Вы видите, кряк Pixelmator Pro 2.0.4 установился и запустился без каких-либо проблем. У знакомого программа не вылетала, но он всего то и сделал, что чуть подкрасил фото моего кролика, которым я с ним поделился, поэтому особо это не учитывайте.

Примеры выше я привожу исключительно в целях демонстрации Вам сути вопроса. Так уж вышло, что для знакомого важно иметь возможность пиратства, и он поделился своим опытом. Ссылок на пиратские программы тут не будет, т.к. сам я категорически осуждаю пиратство, комментарии со ссылками на пиратский контент я также буду удалять. Напомню: это просто отзыв знакомого, верить или нет – дело Ваше. Надеюсь, дисклеймер понят.

Узнать, работает ли Ваша программа нативно на Apple Silicon, можете на сайте isapplesiliconready.com. Там собрана приличная база программ, указываются версии, с которой программы уже начали работать нативно на Apple Silicon, а если ещё не начали – работают ли через Rosetta 2 и есть ли с ними проблемы.

Также Вы во многих случаях Вы можете найти нативную альтернативу Вашему ПО. К примеру, Oracle ещё не выпустили свою версию Java для Mac с Apple Silicon, но Вы можете скачать Zulu сборку OpenJDK от Azul специально для Mac с Apple Silicon.



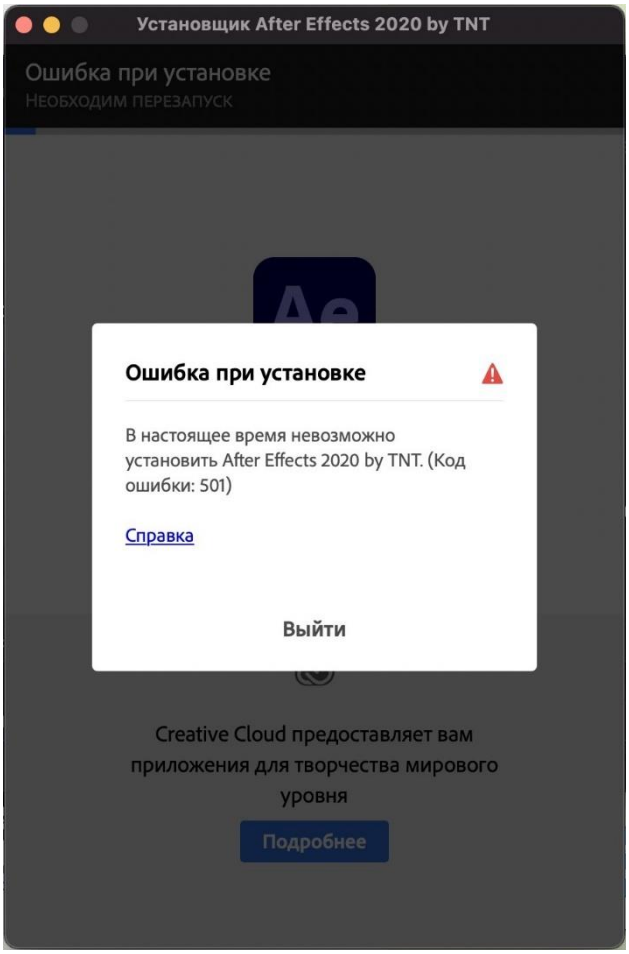
Скриншот 188. Macbook Pro 13 2020 (Apple M1, 20W). Нативный Microsoft Word на Apple Silicon.

Я часть этой статьи писал на Macbook Pro с Apple M1 в Word и особых проблем не испытывал в т.ч. со сторонним расширением Easy Code Formatter. Ничего не вылетало, не тормозило, работать было комфортно с такой плавностью и совершенно абсурдной скоростью в сравнении с моим Xiaomi. Однако я привык к горячим клавишам, используемым в Word в Windows, и они несколько отличаются от таковых в macOS, но если бы у меня было больше времени, думаю, я бы спокойно весь текст написал на Macbook, если вырубить на нём TouchBar, который меня дико бесит (об этом больше информации дам в главе с субъективными впечатлениями).

9.2. ПО под macOS на Intel.

Несколько запоздало эту часть статьи пишу, т.к. многие программы, которые я хотел тут упомянуть, уже вышли в нативном виде для Apple Silicon. К таким отнесу Word, Visual Studio Code, Adobe Premiere Pro и другие. В целом если Rosetta справляется с трансляцией программы, т.е. если она не содержит в себе AVX код, если программа не использует для рендера API OpenGL, а задействует API Metal, хотя бы через MoltenVK, проблем у Вас особых быть не должно. Я этот вопрос рассматривал в главе 2.4.

А ещё нюансы есть? Да, есть. Та же беда с кряками. Сам я не проверял, это всё - информация от знакомого с его экземпляром Macbook на Apple M1.



Скриншот 189. Невозможность установки крякнутой версии Adobe Premiere Pro на Mac с Apple Silicon.

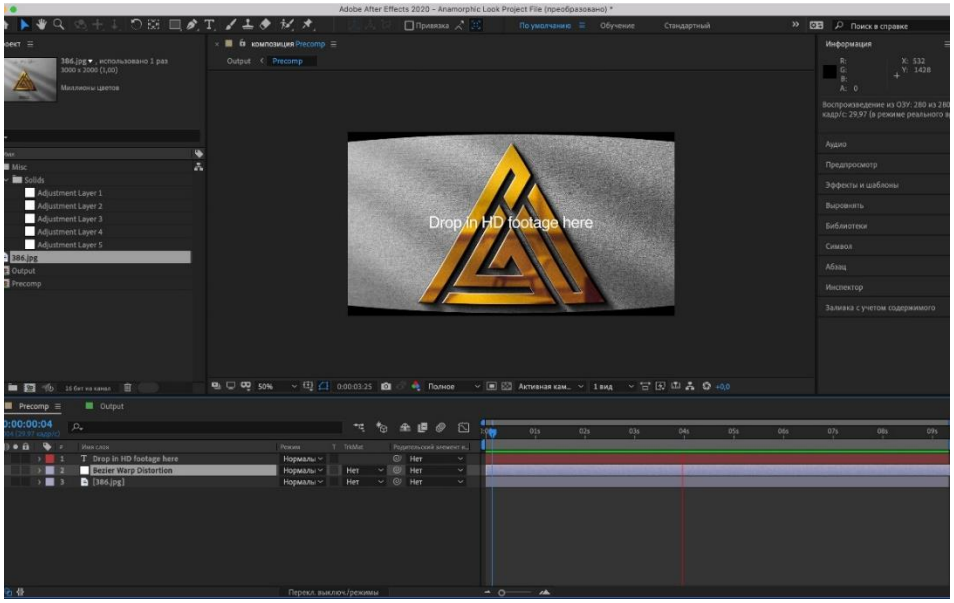
На момент, когда я пишу эту часть статьи, Adobe Premiere Pro ещё не вышел нативно под Apple Silicon. Весь пакет программ Adobe, который знакомый пробовал, включая Adobe Illustrator, Adobe Premiere Pro, Adobe Media Encoder, Adobe Photoshop (теперь уже имеет версию под Apple Silicon наряду с Davinci Resolve), и т.д. имел проблемы на этапе установки с кряком.

В чём может быть причина? Есть у меня одна догадка.

Дело в том, что Rosetta 2 не только может транслировать приложения в момент их первого запуска, но и может транслировать их ещё до того, как Вы решитесь их запустить, чтобы к моменту первого запуска приложение было уже транслировано и не приходилось ждать перевода машинного кода для одной архитектуры в аналогичный для другой. Трансляция в момент запуска зовётся Just-in-time translation, в то время как заблаговременная трансляция x86_64 приложений в ARM64 ещё до того, как Вы решили их запустить, зовётся Ahead-of-time translation. Трансляция приложения может быть произведена, например, ещё на этапе его установки. И это сулит определённые проблемы. Дело в том, что, как я уже упоминал, кряки/патчи работают таким образом, что находят нужный машинный код и подменяют его другим, либо просто вырезают. А как подменить или удалить нужный машинный код, который был транслирован в другую архитектуру? Кряки либо не срабатывают, либо просто ломают программы (в том смысле, что они больше не работают, а не в том, что они становятся взломанными).

Не со всеми крякнутыми программами есть проблемы, тут как повезёт.

Описанное выше – лишь опыт знакомого, у которого имеется на руках Macbook с Apple M1. Верить или нет – дело ваше. Напомню, что сам я крайне осуждаю пиратство, ссылок на пиратские программы в статье вы не найдёте, и комментарии с такими ссылками я буду удалять. Не одобряю.



Скриншот 190. Macbook Pro 13 2020 (Apple M1). Работа Adobe After Effects через Rosetta 2.

Я установил программы Adobe (Illustrator, Photoshop, After Effects) в пробном режиме (доступен в Adobe Creative Cloud при установке Adobe XD Design), также я установил Cinema4D в пробном (trial) режиме.

Я проверил, запускаются ли в программах следующие проекты:

1. Cinema 4D - Pipes;
2. Adobe Photoshop – mockup with gold logo;
3. Adobe Illustrator – wild animal poster wolf sketch florat decor;
4. Adobe After Effects – create anamorphic look;

Везде всё ок, проблем не ощутил, хотя я просто вертел проекты в этих программах, т.е. я ничего особого там не делал, т.к. опыта работы с ними нет.

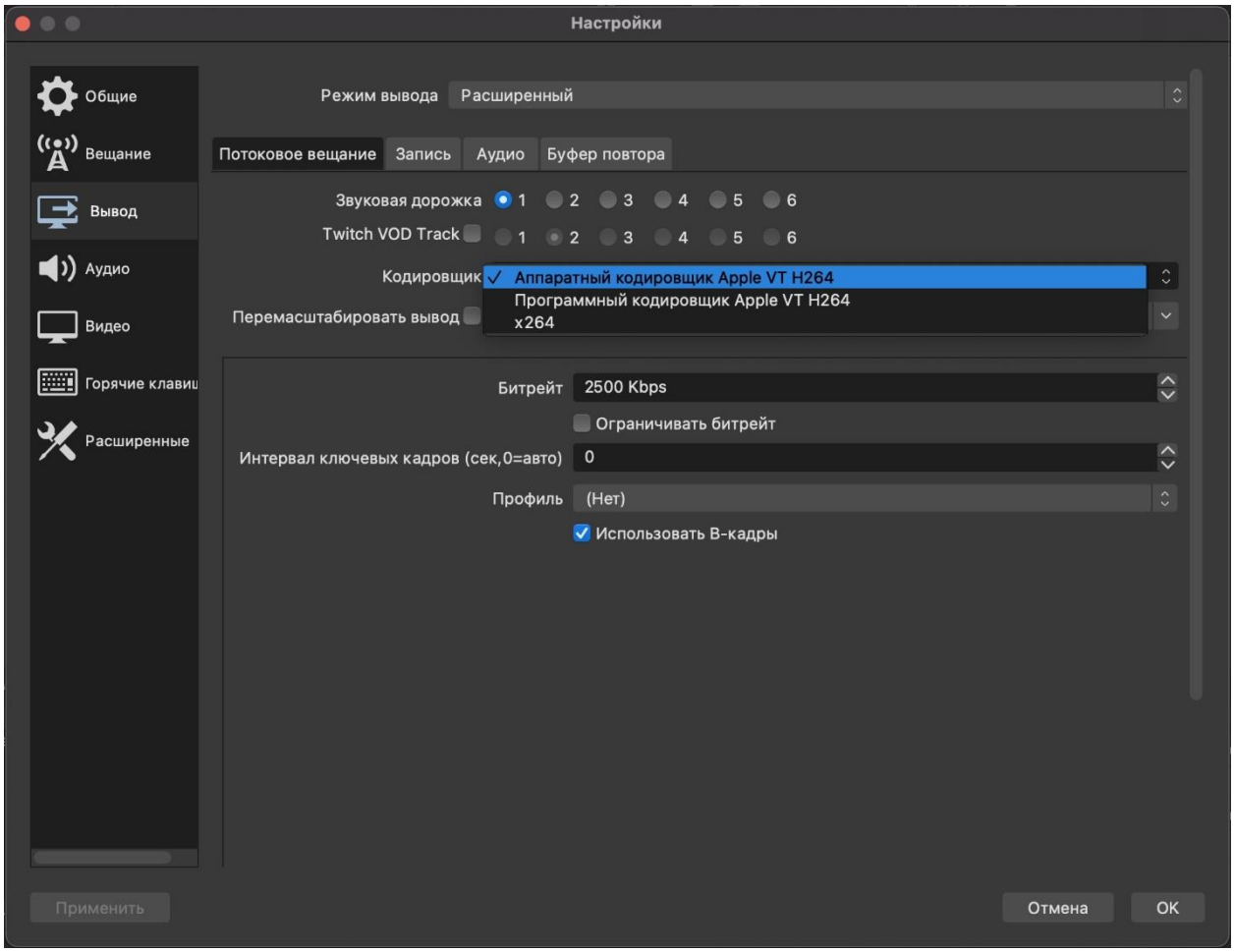
Однако нюансы всё ещё остаются.

Помните, в главе 2.8 я писал про отсутствующее аппаратное ускорение кодирования в формате H.264 (AVC) в OBS? У этой проблемы есть 2 возможных решения: компиляция OBS из исходного кода под Apple Silicon со всеми необходимыми правками, или же замена одной подключаемой библиотеки в папке программы: OBS.app/Contents/Plugins/mac-vth264.so.

Ранее в теме с описанием проблемы на GitHub имелась уже собранная пользователем SharkyRawr из исходного кода нативная версия OBS под Apple Silicon, но у неё позднее выявили ряд проблем, в т.ч. неработающую виртуальную камеру, и на данный момент ссылка в сообщении недействительна, так что ту сборку загрузить не удастся.

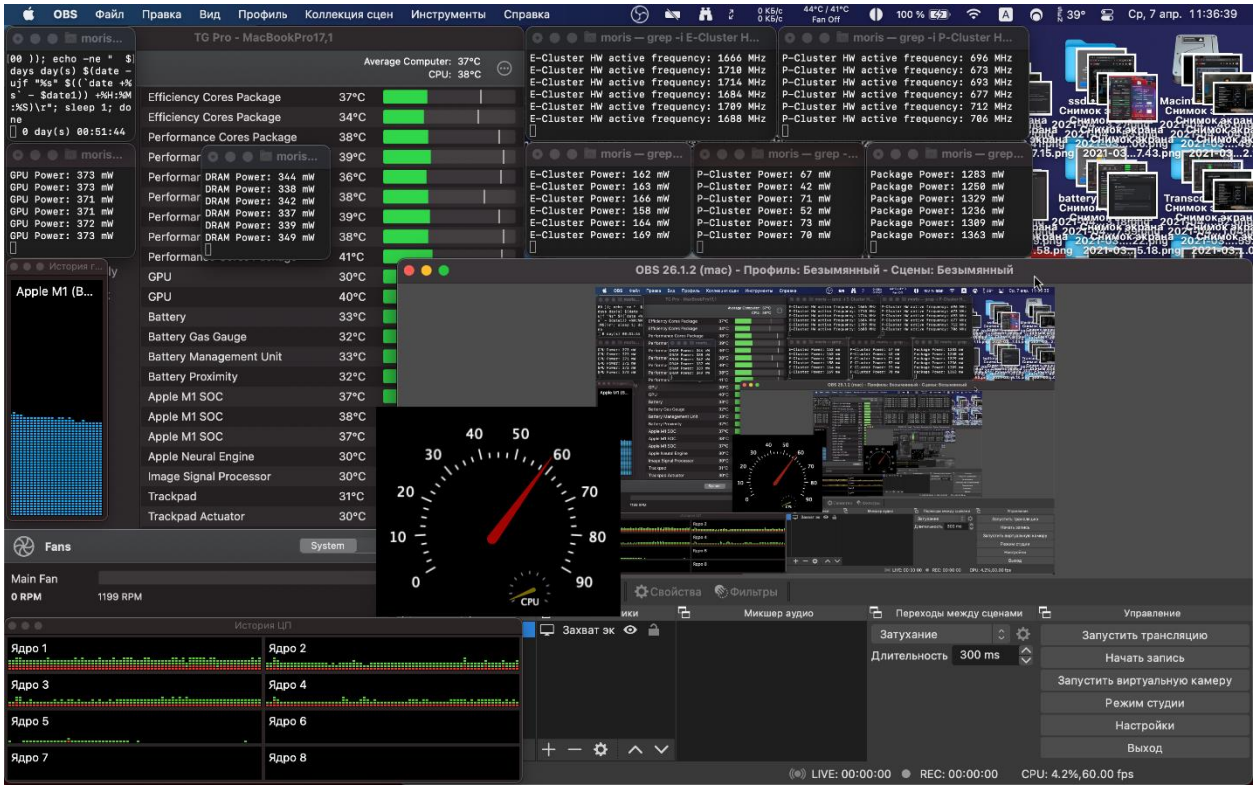
У варианта с заменой файла .so (подключаемой библиотеки) в папке программы также имеется один очевидный недостаток: при установке ограничения битрейта в настройках, OBS не записывает и не стримит аудио.

Но в целом то это рабочее решение? Думаю, да.



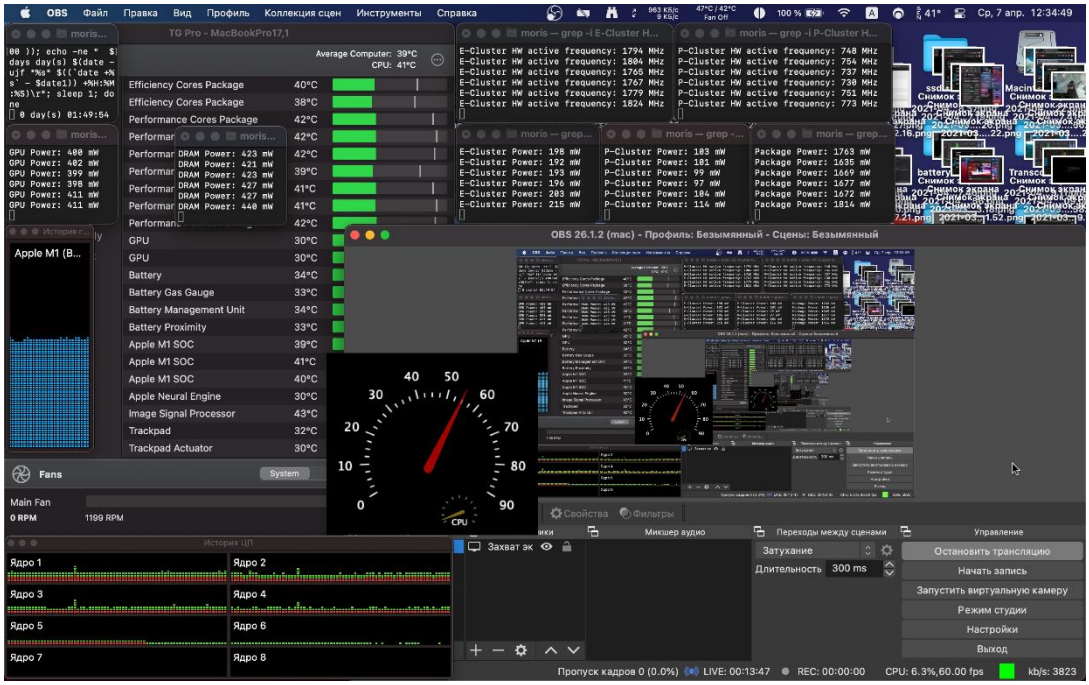
Скриншот 191. Активированная возможность стриминга и записи с помощью аппаратных блоков кодирования видео.

После активации возможности кодирования видео аппаратными блоками предлагаю сравнить потребление энергии «до» и «после».



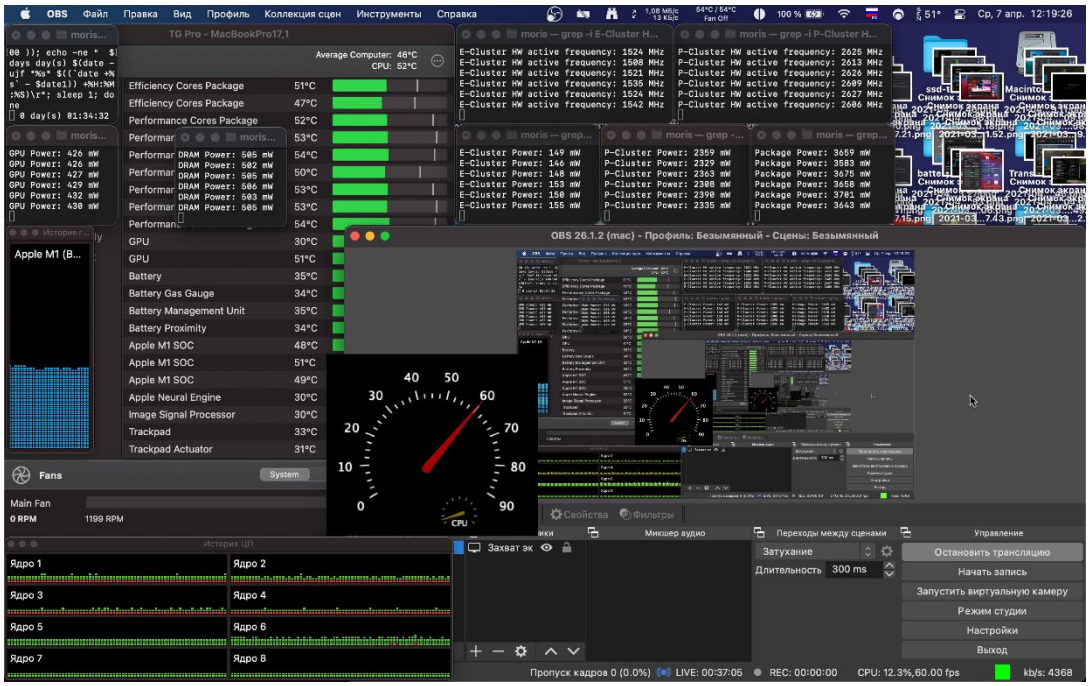
Скриншот 192. Потребление энергии при запущенном OBS в простое, т.е. без запущенной трансляции или записи видео.

Сперва проверили то, сколько SoC Apple M1 потребляет энергии с запущенным OBS без активной записи видео или трансляции: около 1.3 Ватт.



Скриншот 193. Потреблении энергии при запущенном OBS. Стрим видео на YouTube через ReStream.io. H.264 VT HW.

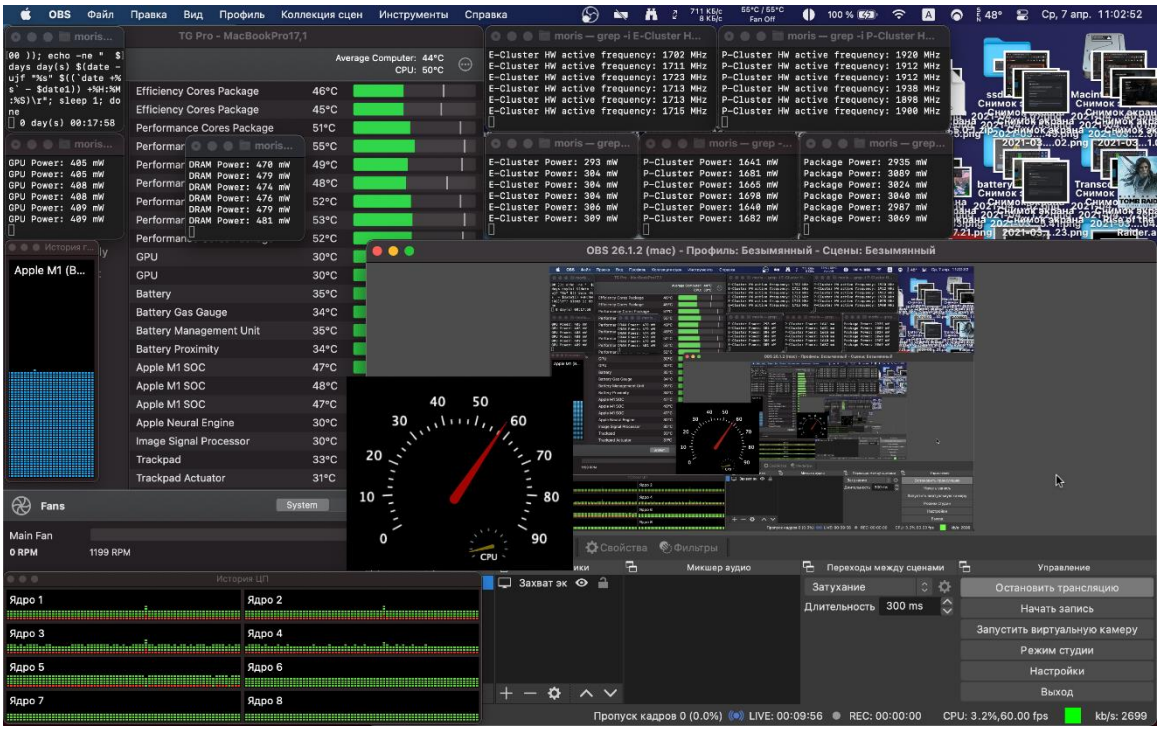
Если стримить на YouTube с использованием аппаратных блоков кодирования в разрешении 1440x900, SoC Apple M1 будет потреблять 1.7 Ватт в среднем. Т.е. с простоем составила всего 0.4 Ватт. Я устанавливал изначально битрейт в 10 Мбит, но он по итогу составил 4 Мбита, поэтому для возможности сопоставления результатов мы выставили битрейт в 4 Мбита и для задачи кодировании центральным процессором.



Скриншот 194. Потреблении энергии при запущенном OBS. Стрим видео на YouTube через ReStream.io. x265.

Кодировщик x265 на ЦП существенно повысил нагрузку на блок производительных ядер в Apple M1. Общее энергопотребление SoC составило уже 3.7 Ватт в среднем. Это на 2.4 Ватта больше, чем в простое и на 2 Ватта (или в 5 раз) больше, чем при использовании аппаратных блоков.

Настройки для кодировщика x265 были установлены следующие: CBR (постоянный, не переменный) битрейт 4 Мбит/сек и предустановка veryfast.



Скриншот 195. Потреблении энергии при запущенном OBS. Стрим видео на YouTube через ReStream.io. H.264 VT SW.

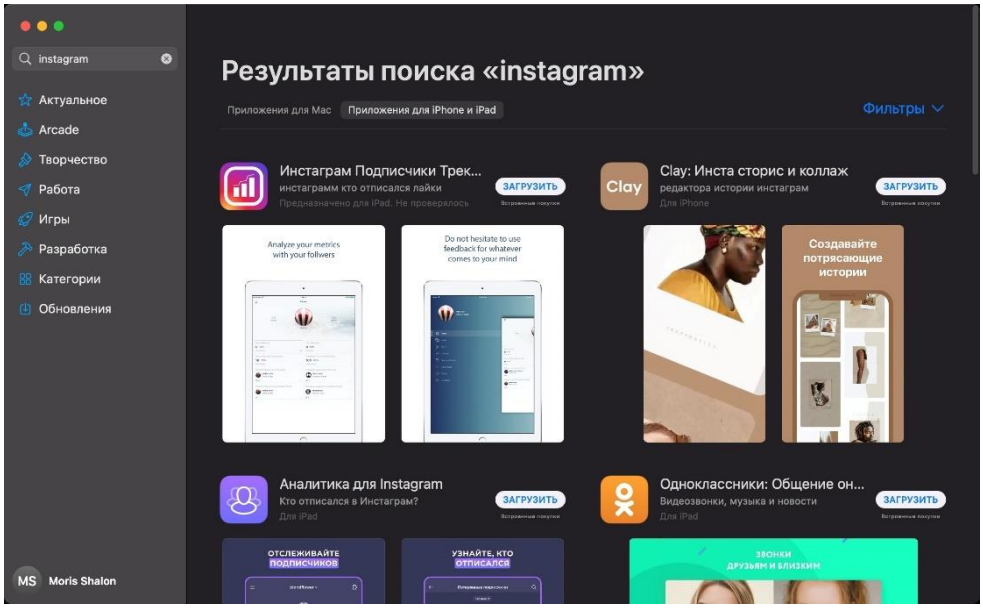
Если задействовать программное кодирование видео в H.264 по API VideoToolBox (или, иначе говоря, кодирование на центральном процессоре), то битрейт почему-то будет ограничен лишь до 2.7 Мбит/сек, т.е. выше этого значения он не поднимется. Потребление энергии в среднем при этом составит 3 Ватта, что на 1.7 Ватт больше, чем в простое, или на 1.3 Ватта (в 3 раза) больше, чем при использовании аппаратных блоков кодирования видео.

Трансляцию все 3 раза вёл в цветовом формате NV12 (709) и в 60 FPS.

К чему это всё? К тому, что часть ПО сейчас работает неполноценно из-за палок, которые Apple вставила в колёса разработчикам изменениями в API. Ваши варианты: ждать адаптации программ от разработчиков, пытаться своими силами заставить всё работать как Вам надо, ждать ещё полгода до адаптации всего перед покупкой Mac, или отказаться от покупки Mac на M1.

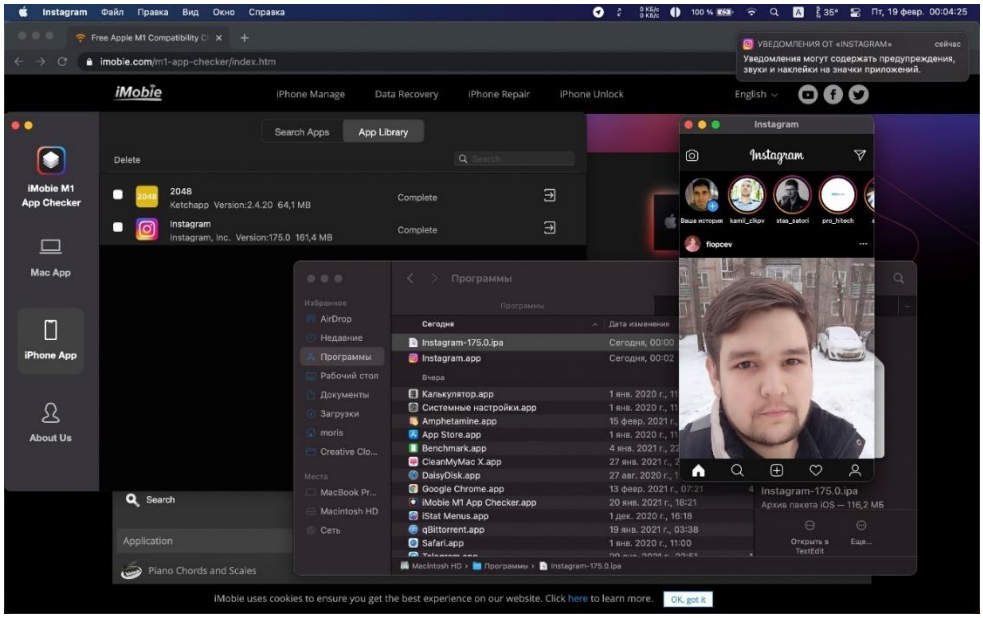
9.3. ПО под iOS / iPadOS.

Тут всё просто: скачиваете себе приложение из AppStore. У iOS/iPadOS и macOS аппаратов ведь сейчас одна и та же архитектура CPU используется, и Операционные Системы у них родственные. Всё должно работать с завода.



Скриншот 196. Отсутствие возможности загрузить приложение Instagram из AppStore на Mac.

Но как быть, если разработчик ограничил возможность установки своего iOS приложения на компьютеры Mac с SoC Apple Silicon? Например, хотите Вы себе загрузить приложение Instagram для iOS на свой Mac, а в AppStore оно не доступно. Есть ли решение?



Скриншот 197. Установка Instagram через iMobile M1 App Checker.

Да, и решение довольно плёвое, если у вас на руках есть аппарат с iOS/iPadOS, но я не могу ручаться за сохранность Ваших данных при

использовании данной утилиты. Загружаем себе iMobile M1 App Checker, авторизуемся в нём тем же аккаунтом, которым загрузили себе Instagram на iOS/iPadOS аппарат и скачиваем Instagram, привязанный к нашему аккаунту.

Тут не важно, насколько исправен Ваш iOS/iPadOS аппарат, как давно Вы им пользовались, тут важно лишь добавить приложение в библиотеку своего аккаунта, т.к. iMobile M1 App Checker и его аналоги умеют скачивать только приложения из Вашей библиотеки.

Логика та же, что и при установке старых версий приложений на iOS аппараты, которые уже давно не поддерживаются (например, iPhone 4). В слишком старых версиях iOS Вы не можете себе даже загрузить старую версию ПО, которое было ещё совместимо ним (например, с iOS 7), поэтому в случае, к примеру. с iPhone 4, приходится сперва добавлять к себе в аккаунт приложение на более свежем аппарате, а потом уже, при совместимости старой версии этого приложения с iOS 7, Вы имеете возможность скачать себе данное приложение на старом iPhone.

Вот тут ровно то же самое, только нет требования к версии Вашей ОС (11.0, 11.1 или 11.2, не важно), и приходится использовать стороннее решение в виде iMobile M1 App Checker или его аналога для загрузки приложения, привязанного к Вашему аккаунту в AppStore.

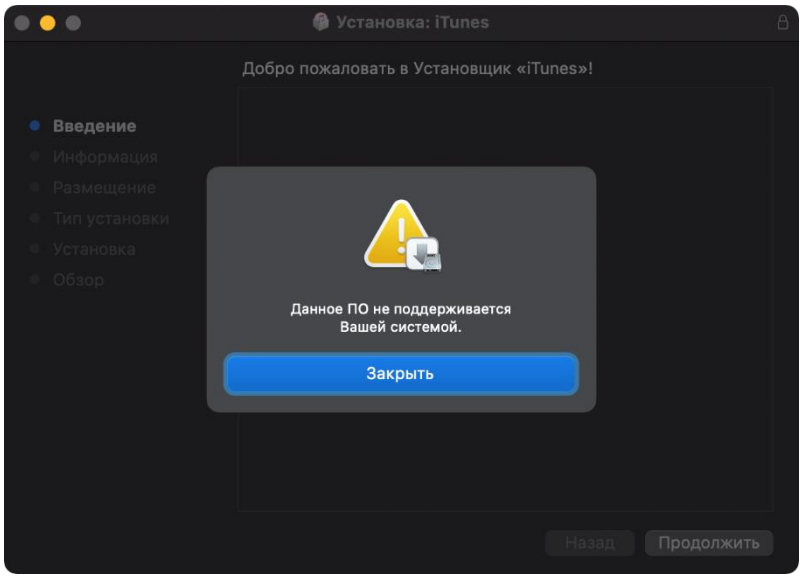
На Ваш страх и риск, в общем. В добавок, если работает некорректно, Вы за это разработчику ничего не предъявите, т.к. приложение для iOS, не доступное в AppStore для Mac, не обязано даже в принципе работать на Mac. Например, Вы можете загрузить себе TikTok и Instagram, и они, по крайней мере, будут работать, а вот игры по типу Super Mecha Champions – нет.

И возникает вопрос: а как быть, если iPhone/iPad/iPod под рукой нет, и, если вообще не являетесь пользователем других гаджетов Apple, кроме компьютеров Mac? Вот я, к примеру, пользуюсь смартфонов Google Pixel 2 XL в качестве основного смартфона и, хоть у меня и лежит старенький iPhone 5S, я им не пользуюсь, т.к. он неисправен. Как быть в таком случае?

Тут на помощь приходит старый добрый iTunes от самой компании Apple. Если помните, то в iTunes до версии 12.7 была возможность покупки

приложений (в т.ч. и добавления к себе в аккаунт бесплатных приложений), и была возможность загрузки установочных .ipa файлов на компьютеры с macOS и Windows для их дальнейшей синхронизации с iOS аппаратами.

Версию 12.6.5.3, последнюю, ещё имеющую возможность загрузки .ipa файлов, можно скачать с официального сайта Apple по ссылкам вот тут.



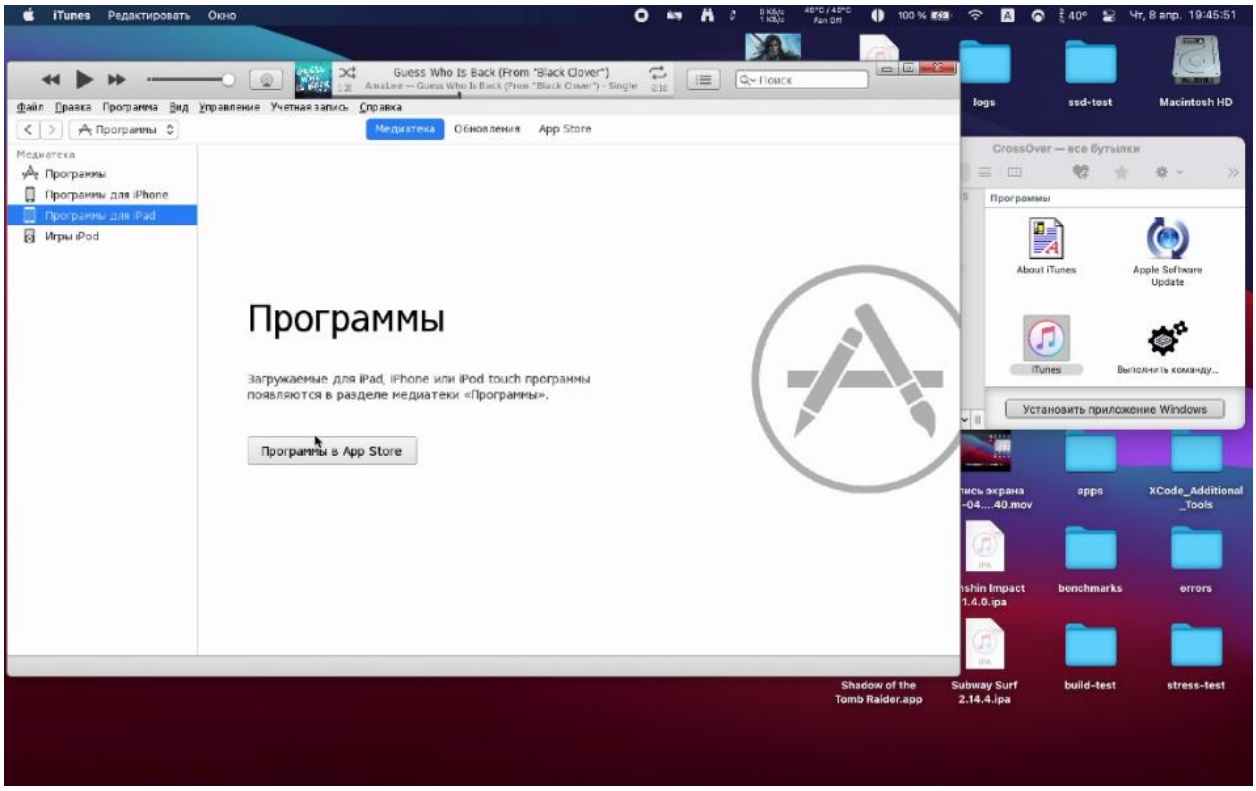
Скриншот 198. Ошибка при установке iTunes 12.6.5.3 на Macbook Pro с SoC Apple M1 в macOS 11.2 Big Sur.

Однако тут есть нюанс: Вы можете установить iTunes версии 12.6.5.3 только на компьютер под управлением ОС Windows, будь то 32-битный или 64-битный компьютер. macOS же, начиная с 10.14 Mojave, не совместима с iTunes версии 12.6.5.3 (по крайней мере, так пишут Apple, сам я на версии 10.14 не проверял iTunes 12.6.5.3). На macOS 11 Big Sur он точно не работает.

С Windows всё просто: ставите себе iTunes 12.6.5.3, скачиваете .ipa файл для iOS/iPadOS из App Store и отправляете его себе на Macbook с M1. Но как быть, если компьютера с Windows нет под рукой? Скажу, Вы смелый, если после чтения этой статьи купили себе Mac на M1, не оставив запаски.

Это просто моё мнение: Macbook на M1 стоит взять только в том случае, если у Вас остаётся ещё хоть один запасной ноутбук, т.к. с ПО для Windows будут проблемы на M1. Я подробнее раскрою тему в главе 9.6.

Тем не менее, решение есть даже в этой ситуации. Но решение это крайне убогое и уж лучше на ноутбук с Core 2 Duo и 2 ГБ оперативной памяти загрузить iTunes, чем использовать костыли, описанные ниже.



Видео 7. Демонстрация работы iTunes 12.6.5.3 в CrossOver и проверка работоспособности iOS/iPadOS приложений.

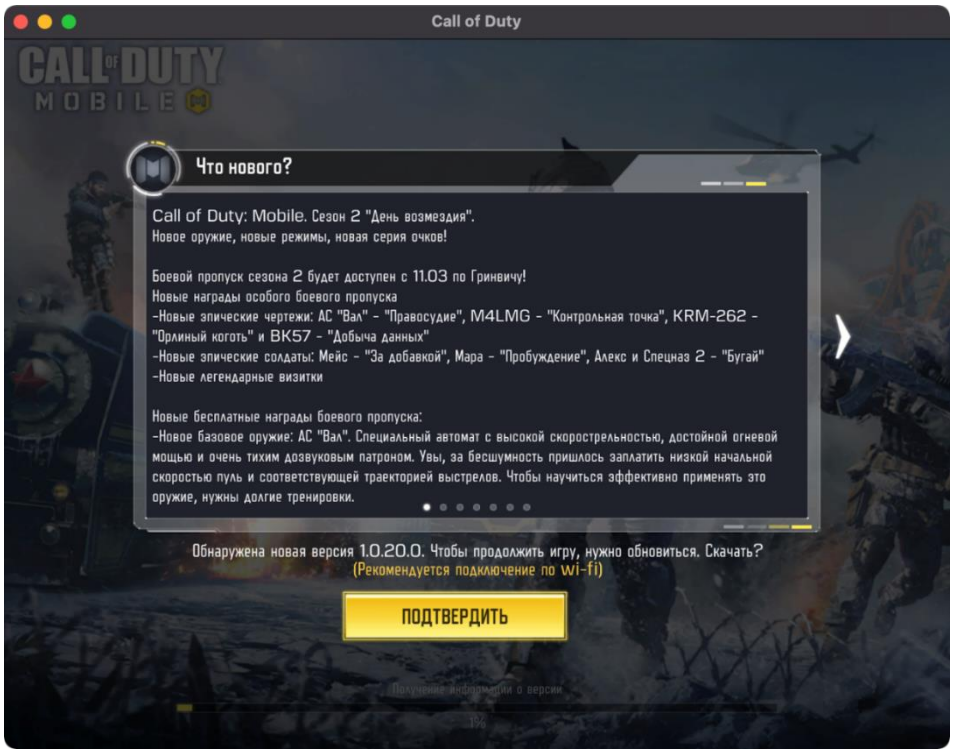
Посмотрите видео. Пробовал iTunes и x86, и x64. Я на 0:58 пытался быстро включить музыку, перемотать её и выключить, но настолько сильно всё тормозит, что на начало воспроизведения ушли пол минуты, и ещё пол минуты на перемотку. На остановку музыки ушли секунд 40. Мне из-за этого на ролик, доступный по скрытой ссылке, автоматически прилетела жалоба по Content ID и оттого на видео висит реклама от правообладателя... М-да.

Фактически, iTunes через CrossOver работает и Вы можете через него загружать себе приложения для iOS/iPadOS на Mac с Apple M1. Но тормозит всё настолько, что лучше уж отдельный комп на винде использовать.

Работают: TikTok (лучше бы не работал, это просто плохо), Instagram, Subway Surf, Among Us, Crash Bandicoot, Genshin Impact и Call of Duty Mobile.

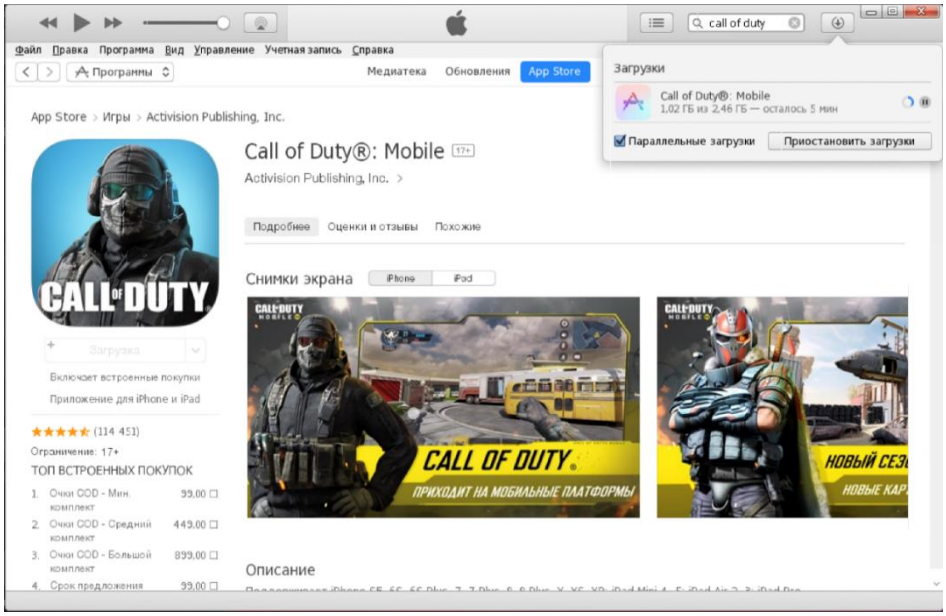
Не работают: PUBG Mobile и Super Mecha Champions.

В общем, что-то с iOS будет работать, а что-то нет. Тут уж как повезёт.



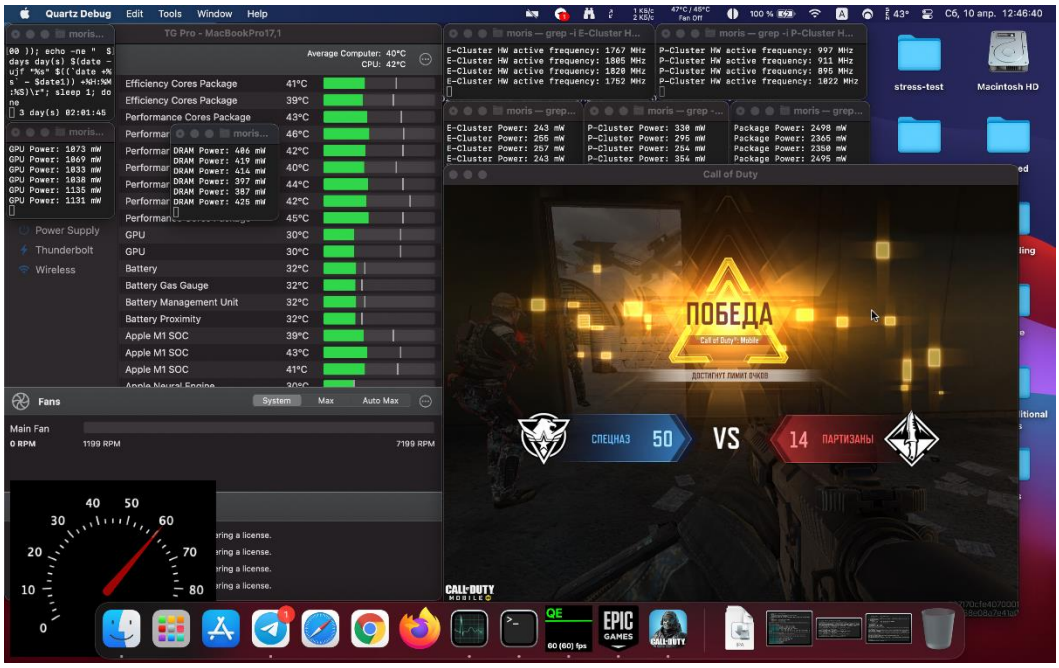
Скриншот 199. Запрос Call of Duty Mobile на обновление.

Есть нюанс: приложения для iOS и iPadOS периодически отказываются работать без обновления. Пример: Call of Duty Mobile. Пока Вы его не обновите, играть Вы не сможете. Если же нажмёте кнопку «подтвердить», Вам перекинет в Mac App Store, но там Вы этой игры не найдёте, т.к. игра для macOS не выпускалась.



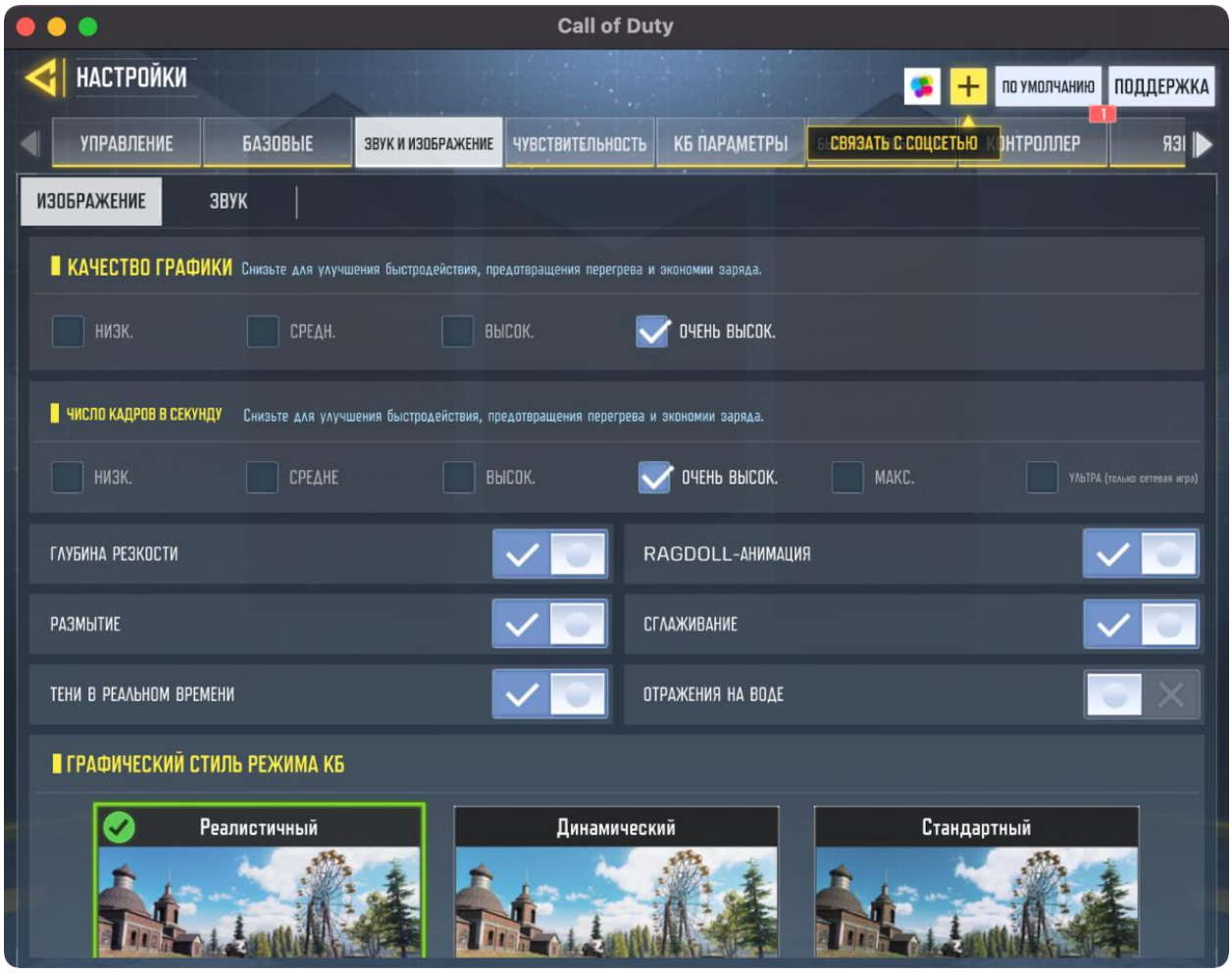
Скриншот 200. Обновление Call of Duty Mobile через iTunes.

В итоге Вам через iTunes придётся каждый раз загружать заново игру. Я не проверял, могут ли скачиваться исключительно обновления, т.е. без загрузки игры заново, т.к. я ранее не оставлял .ipa файлов после установки.

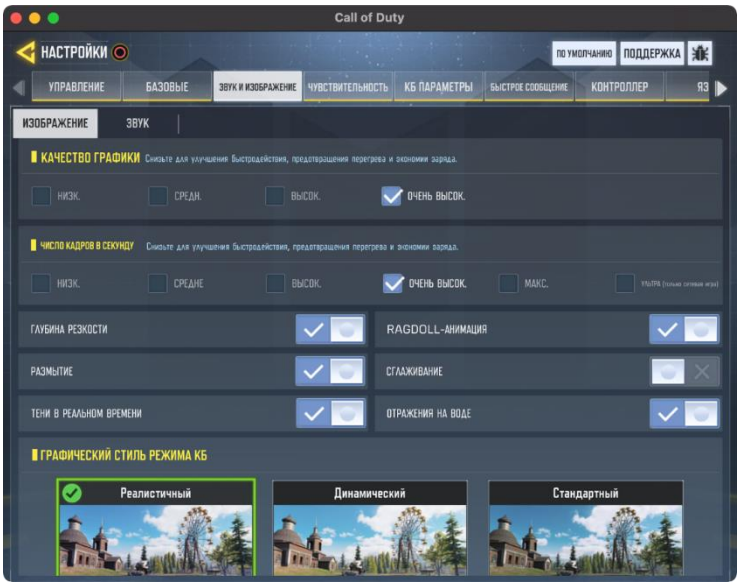


Скриншот 201. Macbook Pro 13 2020 (Apple M1, 20W). Мониторинг FPS в Call of Duty Mobile.

После обновления то всё работает хорошо. И с геймпадом можно нормально играть в Call of Duty Mobile со стабильными 60 FPS на с максимально доступными настройками графики.

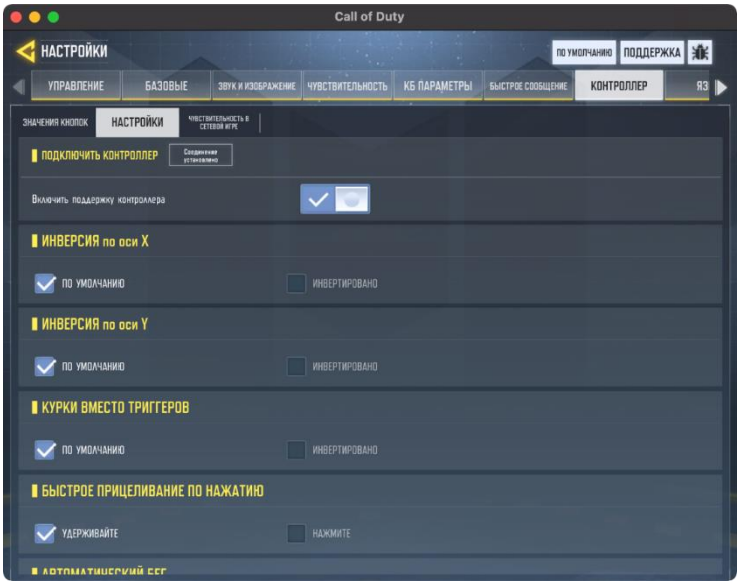


Скриншот 202. Макс. графика со сглаживанием в Call of Duty Mobile на Macbook Pro 13 с Apple M1.



Скриншот 203. Макс графика с отражениями на воде в Call of Duty Mobile на Macbook Pro 13 с Apple M1.

Отражения на воде включить можно, но только отключив сглаживание. Я проверял и со сглаживанием, и с отражениями на воде, в обоих случаях тепловыделение Apple M1 при такой нагрузке не превышает 2.5 Ватта. SoC даже не греется, температура всё время ниже 50 градусов. И кулеры он даже не включает. В общем, тестировать нагрузку в таких играх мы не будем, они ну вот слишком слабо его грузят. M1 их жуёт как семки.



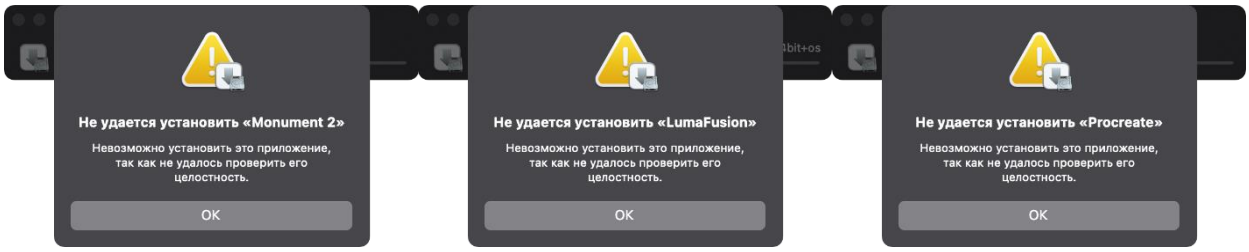
Скриншот 204. Call of Duty Mobile. Отваливается геймпад на Apple M1.

Вот только есть проблема со стабильностью работы с геймпадом. Геймпад у меня то работал в игре, то нет. Он определялся и вверху в настройках геймпада было написано «соединение установлено», но что бы я ни делал с геймпадом, реакции не было в игре. Разве что я видел реакцию на нажатие кнопок Options+Share, тогда возникал запрос на запись экрана.

Опять же, никто не даёт гарантий, что всё будет хорошо работать.

UPDATE: *начиная с macOS 11.3, закрыли возможность установки iOS/iPadOS приложений не из AppStore из .ipa файлов. На AppleGamingWiki описали, как это обойти. Нормальных решения всего 2: снести macOS 11.3+ и накатить 11.2.3, или поставить приложения через PlayCover, который ещё не идеально работает и постепенно допиливается. Ну, куда деваться.*

Стоит ещё открытый вопрос по крякнутым приложениям. Тут также знакомый поделился своим опытом работы с ними на Macbook с Apple M1.



Скриншот 205. Попытка запуска крякнутых .ipa.

По итогу: нет, крякнутые приложения работать не будут. Знакомый пробовал Monument 2, ProCreate и LumaFusion. Последние 2 было бы странно использовать на macOS, когда для первого есть более функциональные альтернативы в виде Adobe Photoshop и Pixelmator Pro 2, а для второго имеется более функциональный аналог Final Cut Pro X от самой Apple.

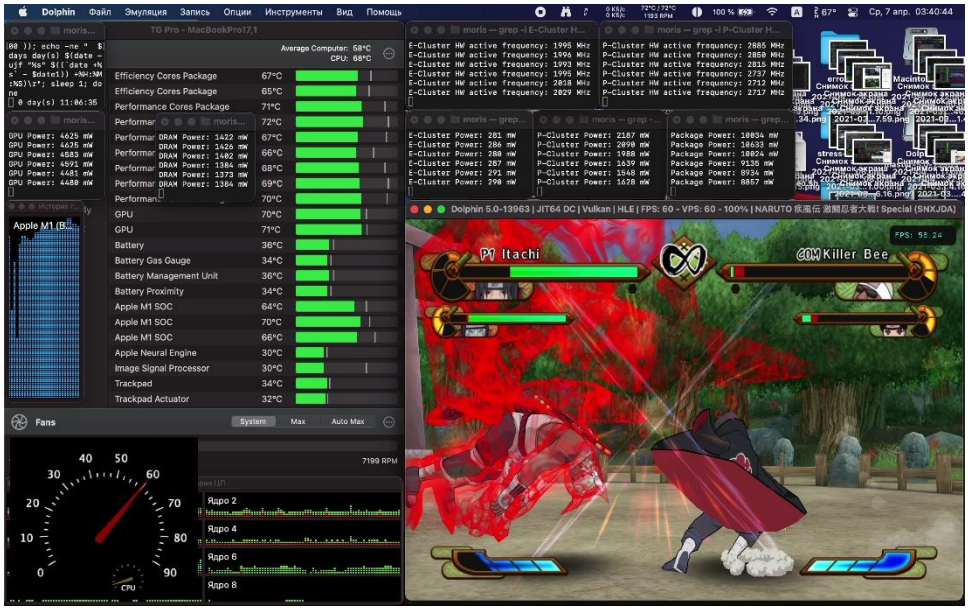
Выше приведён лишь опыт моего знакомого, а сам я устанавливаю только лицензионное ПО и осуждаю пиратство. Но почему с кряками есть проблемы на Mac с Apple Silicon? Есть у меня предположение. Процитирую статью «Your Computer Isn't Yours» от Jeffrey Paul, переведённую на русский язык и опубликованную на habr пользователем Sanctuary_s:

«Оказывается, текущая версия macOS отправляет в Apple хэш (уникальный идентификатор) при запуске каждой программы. Многие люди не были в курсе этого, так как хэш передается незаметно и только при наличии выхода в интернет. А сегодня серверы работали очень медленно и не успевали проверять хэши. Как результат, все приложения не открывались, если имелся выход в интернет.»

macOS анализирует Ваши приложения, и, похоже, проверяет их в том числе по части кряка, оттого и проблемы с .ipa файлами. Решения не нашли.

9.4. Эмуляция других платформ.

В главе 2.4 я упоминал эмулятор Dolphin, который позволяет запускать на Windows, macOS и Linux игры от Nintendo Wii и Nintendo GameCube. Пока я дошёл до написания этой части статьи, он успел обновиться до версии 5.0-13963, тогда как в главе 2.4 я тестировал версию 5.0-13603. Стало ли лучше?



Видео 8. Тест эмулятора Dolphin в игре Naruto Shippuuden Gekitou Ninja Taisen Special.

Нет, в игры от Nintendo GameCube всё также некомфортно играть, как и ранее. Я записал новое видео через QuickTime, чтобы показать Вам геймплей, и в целом всё также некомфортно играть, как и ранее. Тормоза просто жуткие, хотя FPS всегда около 60. Однако, на 28:50 я активировал вертикальную синхронизацию, и картинка стала плавной. На 39:42 я выставил настройки графики: разрешение 1080p, 2x MSAA сглаживание и 2x анизатропная фильтрация. По итогу получил стабильные 60 FPS.

Вот только одна загвоздка: при включении вертикальной синхронизации анимация в игре перестала тормозить, но зато игра начала с тормозами реагировать на нажатие кнопок на моём геймпаде. С моим Sony DualShock 4 при этом нет проблем с играми в других тестах и нет проблем в том же Dolphin без вертикальной синхронизации. В общем, рабочего решения придётся ещё подождать.

Если Вы используете Dolphin, не спешите продавать свой старый компьютер. Другие эмуляторы я не проверял, про них не скажу.

9.5. Виртуализация. Docker.

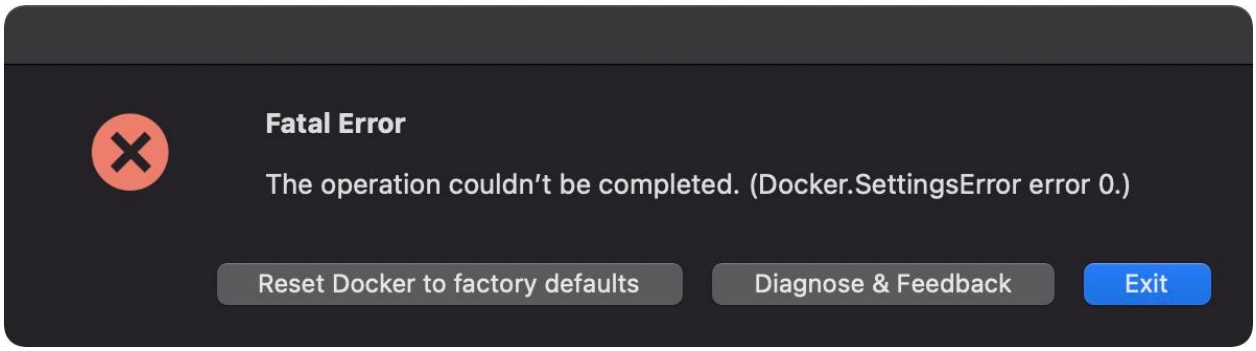
Как дела с виртуализацией, в частности – с Docker? Да нормально. На данный момент он уже имеет версию Docker Desktop 3.1.0 RC 3 от 1 апреля, тогда как я тестировал ещё technical preview 7 (сборка под номером 60984).

У Docker есть 2 значимых нюанса:

Он требует для работы установленный Rosetta 2, т.к. часть библиотек, которые он задействует в работе, всё ещё предназначены для x86_64.

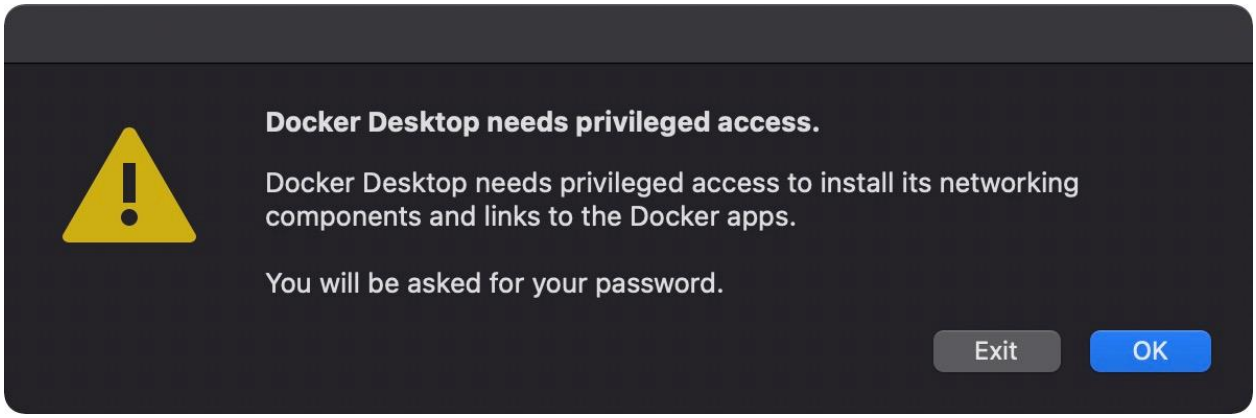
Не все образы доступны для ARM64, поэтому предлагается разработчикам использовать параметр --platform linux/amd64 для запуска образов для Mac с SoC Intel при помощи эмуляции. Не сказать, что это хорошее решение, но лучше, чем ничего.

Подробнее по поводу Docker можете прочитать на официальном сайте.



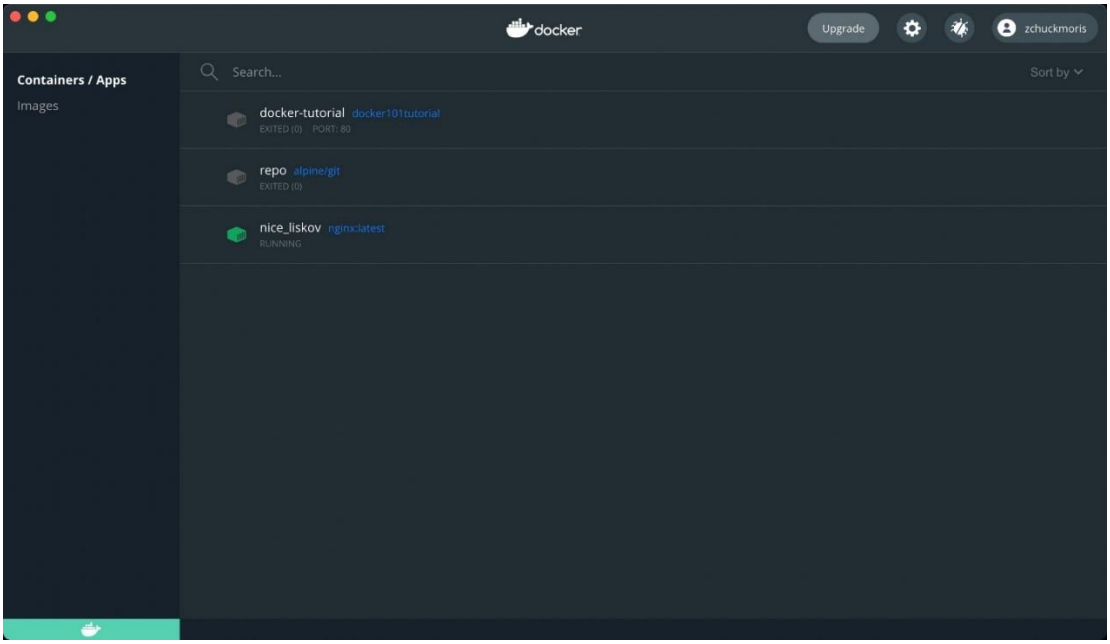
Скриншот 206. Ошибка при запуске Docker Desktop 3.1.0 Technical Preview 7.

Сперва, когда я загрузил Docker, установил его и запустил, я поймал ошибку. Т.к. в Docker ранее у меня ничего ценного не было, ведь я устанавливал на Mac с M1 Docker 1-ый раз, я, не особо задумываясь, нажал на “Reset Docker to factory defaults”.



Скриншот 207. Docker Desktop 3.1.0 Technical Preview 7. Запрос привилегированных прав.

После успешного сброса, Docker запросил привилегированные права, я ему их предоставил, и дальше всё пошло как по маслу по инструктажу.

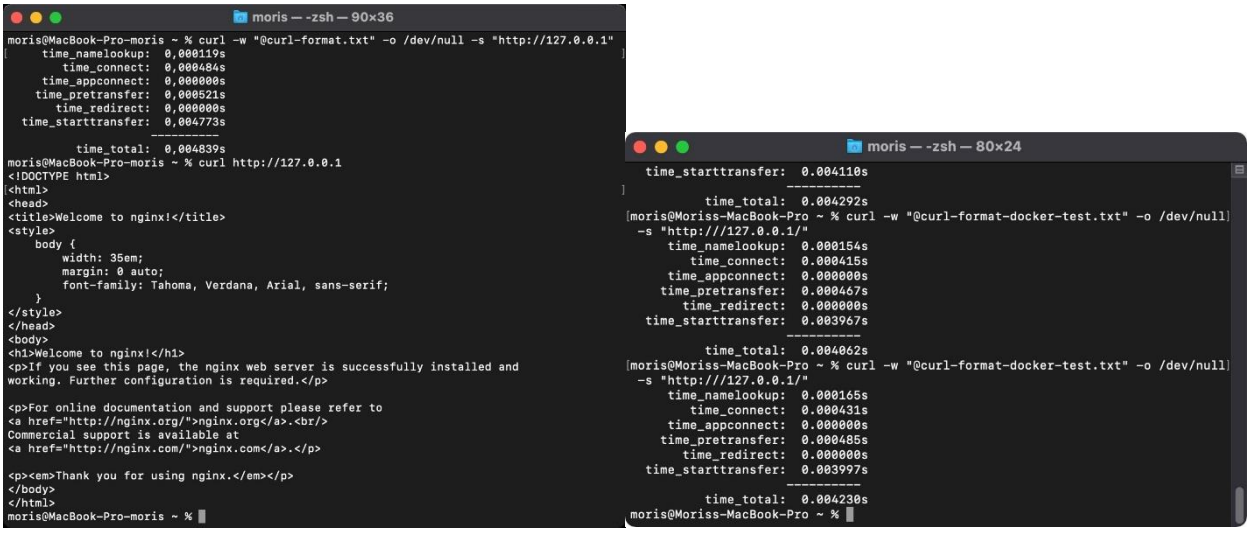


Скриншот 208. Docker успешно установился, nginx также был успешно установлен и запущен.

В общем, Docker установился довольно просто, затем быстро запустил в docker образ docker-tutorial с приветственной страницей на порту 80.

Один человек попросил меня проверить, запускается ли nginx в docker и с какой скоростью парсится стандартная домашняя страница nginx.

Для парсинга я использовал .txt файл с шаблоном и команду отсюда.



Скриншот 209. Скорость парсинга стартовой страницы nginx командой curl. Macbook (M1) vs Xiaomi (i7).

Macbook с M1 справился за 4.8 мс, а Xiaomi с i7 – за 4.2 мс. У обоих меньше 5 мс ушло, так что проблем с Docker лично я не вижу.

Если Вы разработчик, советую ознакомиться с видео «MacBook Air M1 после двух месяцев» на канале Диджитализируй. Тут я многого не расскажу.

9.6. Приложения для Windows.

Если вкратце, тут просто мрак. Всё, забудьте, этого не существует. Проехали, просто переходите к следующей главе. Если Вам нужна возможность запуска Windows-приложений, или не покупайте Mac, или имейте в резерве другой компьютер с ОС Windows на борту.

Можете остаться и прочитать подробнее, что же не так с приложениями для Windows на новом Mac с Apple Silicon. Эта тема будет чуть длиннее предыдущих, а итог я уже описал вкратце выше.

Итак, начнём с того, что Вы на данный момент не сможете нативно запустить здесь Windows 10 через BootCamp в отличие от Mac на Intel. В чём причина? Ведь имеется же Windows 10 on ARM? Почему бы не запустить её?

Причина тут в лицензии на Windows 10 on ARM. Microsoft продают лицензии Windows 10 on ARM только OEM-производителям, т.е. Вы можете купить Windows 10 on ARM только в составе готового решения. Вы не можете просто сходить в магазин, купить флешку Windows 10 on ARM в коробке с цифровым ключом, и в этом первая проблема. Вы ведь не думали, что Microsoft просто позволит Вам пользоваться своей Windows бесплатно. Возможно, просто моё предположение, вторая проблема кроется в том, насколько компьютеры Mac производительнее аналогичных решений на ARM от других производителей. Если компания Microsoft разрешит устанавливать свою Windows на Macbook, который превосходит решения её клиентов, она рискует лишиться этих самых клиентов, так ещё и она сама подсобит таким образом своему конкуренту. В любом случае, одной ли причиной всё ограничивается или же нет, всё сводится к вопросу денег.

Старший вице-президент Apple по разработке программного обеспечения, Крейг Федериги (Craig Federighi), в интервью изданию Ars Technica пояснил, что технически на новых Mac с M1 можно запустить Windows 10 для ARM, которая, в свою очередь, поддерживает x86-приложения. Apple реализовала в M1 все базовые технологии для этого.

Похоже на то. Сам я в M1 не вижу ничего несовместимого с Windows.

Т.е. нет ничего, что мешало бы запуску Windows, кроме вопроса денег.

И тут встаёт вопрос: а что же можно запустить и в каком виде можно заставить приложения для Windows работать на Mac с Apple Silicon?

Варианта всего 2:

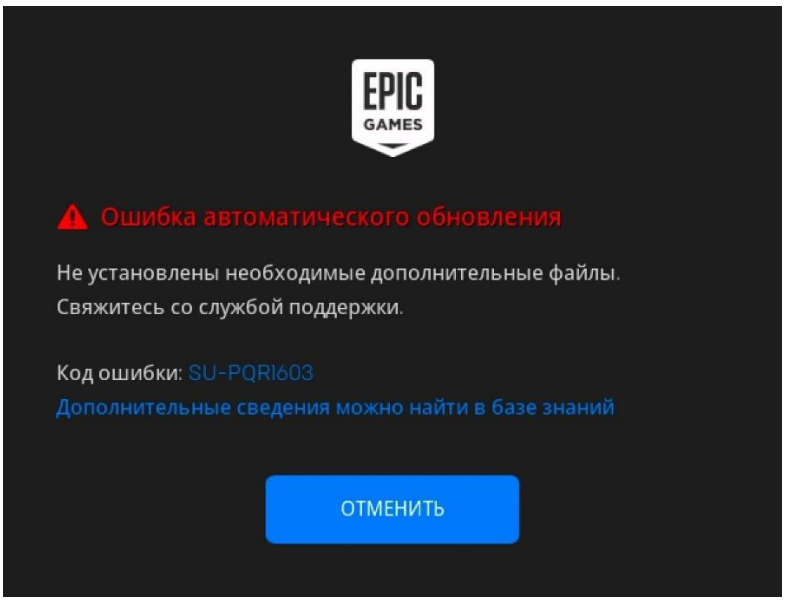
1. Использовать Crossover на базе wine. Если Вы имели опыт с CrossOver, знайте, на M1 всё будет совсем страшно. Тут аж 2 трансляции: x86_64 -> ARM (Rosetta 2) и Windows -> macOS. А для 32-bit приложений ещё и мост задействуется x86 -> x86_64.
2. Использовать виртуальную машину для запуска образа Insider Preview версии Windows 10 on ARM, который Microsoft распространяет бесплатно.

Начнём с 1-го варианта. Спойлер: оба решения не рекомендую никому.

Я читал на reddit о том, что с 13 октября стала доступна версия Crossover 20.0.0, которая привнесла с собой не только поддержку macOS 11 Big Sur, но и поддержку DirectX 11 на macOS. Естественно, данную новость я не оставил без внимания, и загрузил Crossover 20.0.4 на MBP 13 с Apple M1.

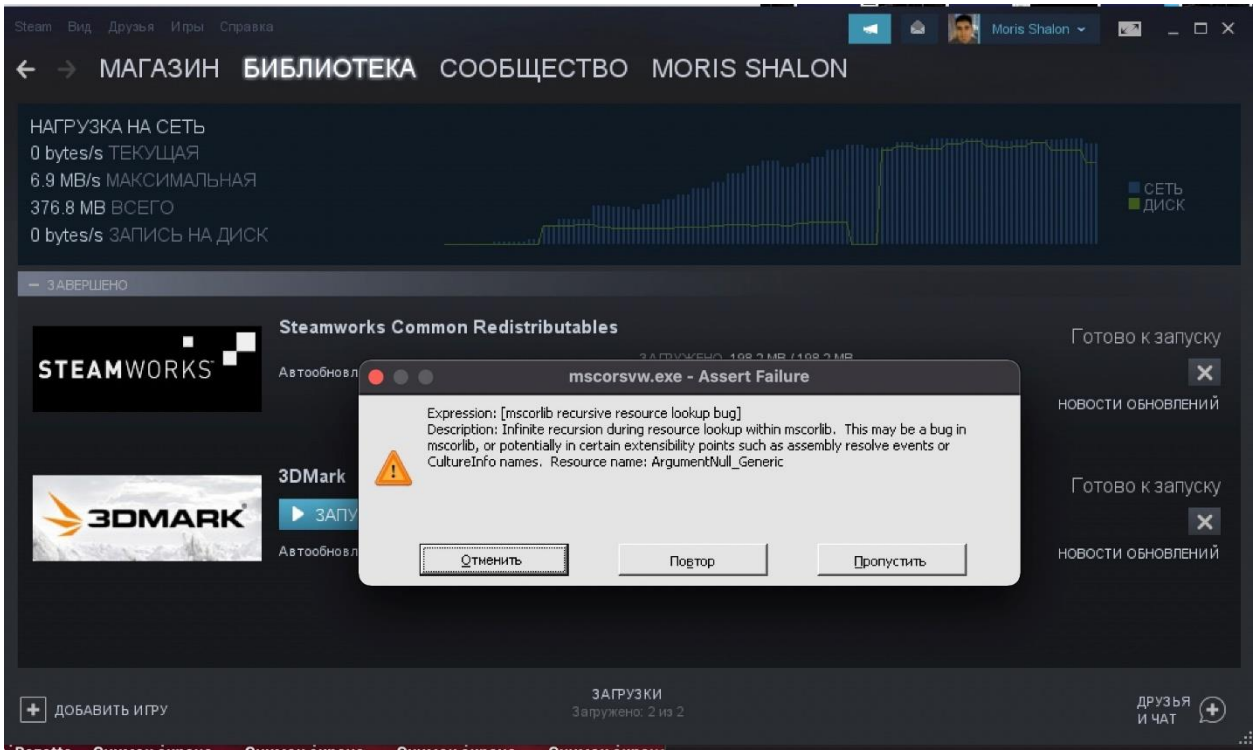
Со списком изменений в каждой версии Crossover можете ознакомиться на сайте codeweavers.com (сайт разработчиков Crossover).

Имеется одна серьёзная загвоздка: никто в codeweavers не заявлял о нативной поддержке Mac с Apple Silicon, пока релиз лишь готовится. Да и даже если это реализуют, нативного софта под ARM на Windows мало, т.ч. в любом случае придётся тут транслировать/эмулировать x86_64, а это больно.



Скриншот 210. Невозможность запуска Epic Games Store в Crossover.

Что и требовалось доказать: нет никаких гарантий, что ПО, запущенное в CrossOver, будет исправно работать. Тот же Epic Games Store мне не удалось запустить несмотря на то, что я установил все требуемые для этого компоненты. Я несколько часов пробовал бутылки с разными настройками и разным набором предустановленных компонентов в CrossOver, но никак. А вот iTunes в главе 9.3 заработал без лишней возни, но заработал ужасно.



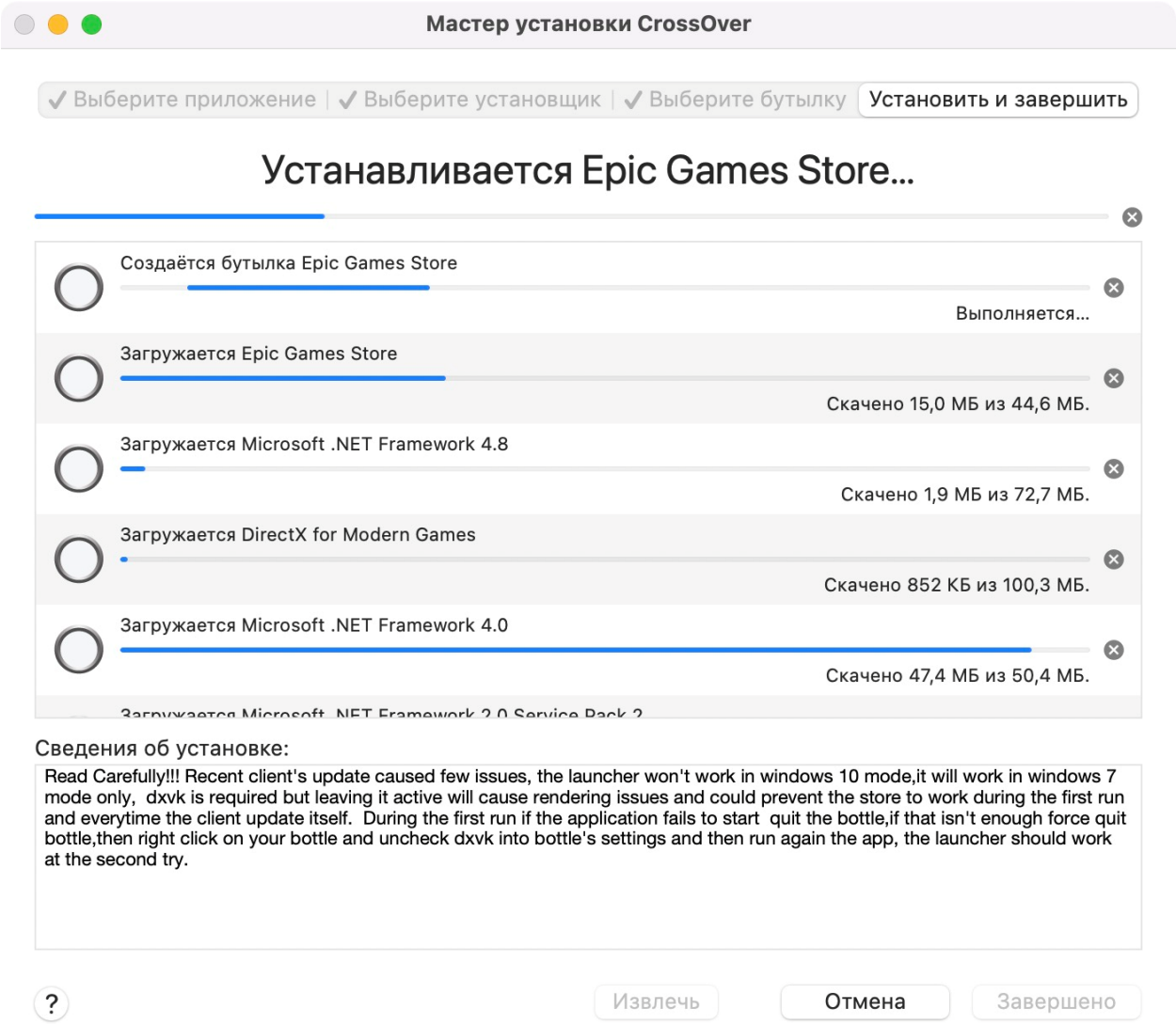
Скриншот 211. Ошибка в Steam в CrossOver.

У меня подобного рода ошибки были со всем софтом, как-либо затрагивающим графику, который я пытался запустить через CrossOver.



Видео 9. Попытка запуска теста в 3DMark в CrossOver.

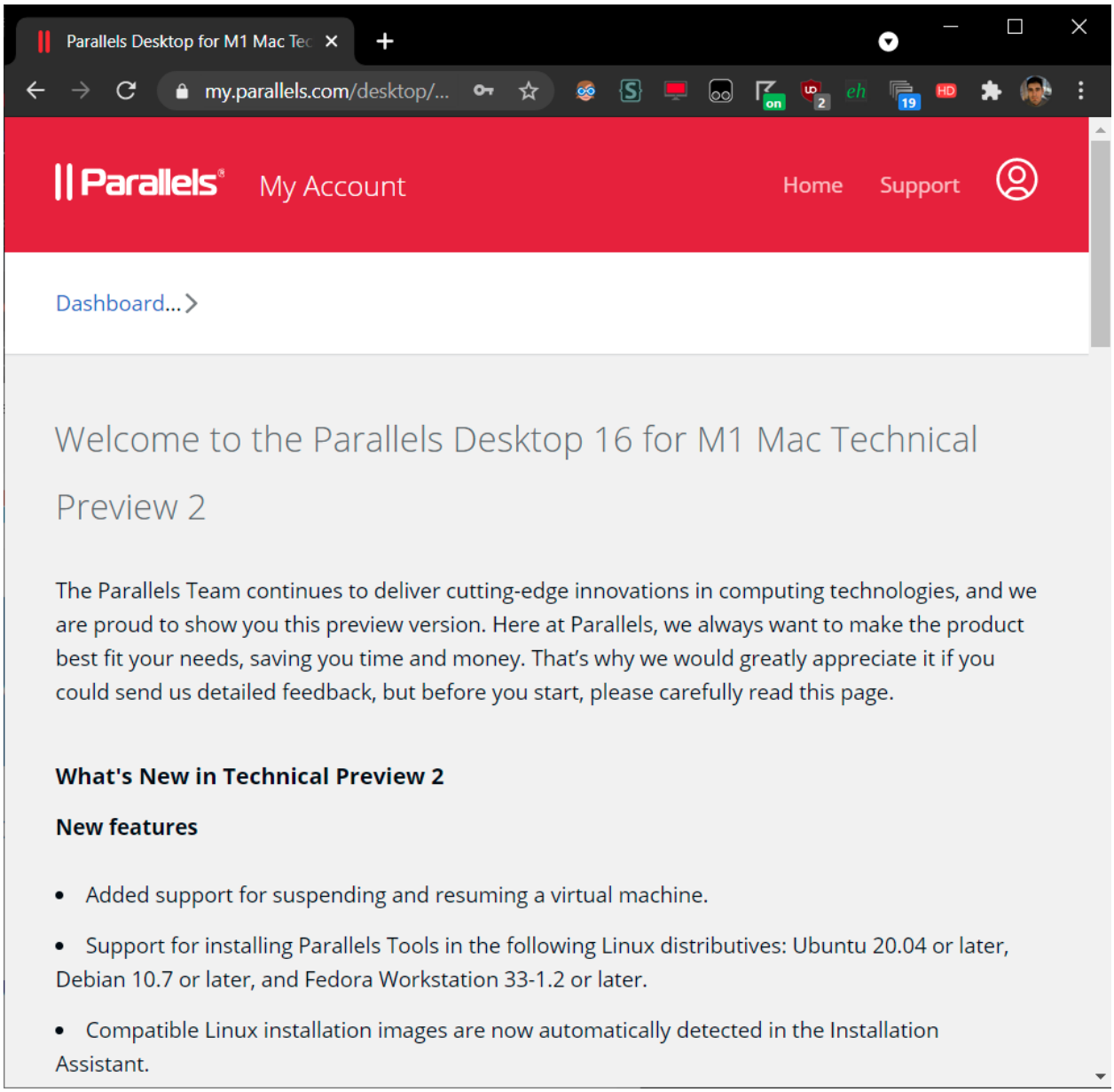
Просто посмотрите выше видео, где я пытаюсь запустить тесты DirectX 11 3DMark, установленный через Steam в CrossOver. 5 минут боли для Вас и несколько часов боли для меня. Я переустанавливал 3DMark несколько раз, пробовал всё с разными настройками в CrossOver, но результат всегда один: он запускается целую вечность, а после запуска ещё и тормозит до жути. На M1 с такими-то тормозами не будет никакого прока от поддержки DirectX 11.



Скриншот 212. Создание новой бутылки в CrossOver.

Повторюсь: я перепробовал все варианты, и Вы сами видели в главе 9.3, как тут работает iTunes. Да, что-то через CrossOver работает, но работает ужасно. Может быть, он будет вполне неплох, когда выйдет версия под Apple Silicon, но сегодня я не рекомендую его рассматривать как решение для какой бы то ни было задачи. Нужна винда? Не спешите избавляться от компьютера на Windows, и настройте на нём удалёнку для доступа вне дома.

Вы можете быть со мной не согласны, тогда ещё раз пересмотрите видео из главы 9.3, где я запускаю старую версию iTunes на новом Macbook с целью загрузки iOS/iPadOS приложения, не доступных в App Store для Mac на Apple Silicon. iTunes через CrossOver из-за двойной трансляции (x86_64 -> ARM и Windows -> macOS) тормозит просто безбожно. Совсем не вариант.



Скриншот 213. Информация о Parallels Desktop 16 Technical Preview 2 с сайта разработчиков Parallels.

Следующий вариант: Parallels Desktop. На данный момент для Apple Silicon он доступен на официальном сайте только в Technical Preview версии. Вы можете зарегистрироваться на сайте Parallels и скачать себе Technical Preview версию бесплатно, а затем активировать его ключом, который там опубликован. Ключ к Вашему аккаунту привязан не будет, так что не получится так на будущее получить бесплатно стабильный Parallels.

Windows 10 on ARM Insider Preview

With Windows 10 on ARM Insider Preview builds, you can create 64-bit ARM (ARM64) VMs in Hyper-V on Windows 10 ARM-based PCs. Creating ARM64 VMs is not supported on x64 hardware.

ARM64 VMs are only supported on devices that meet the pre-requisites:

- Windows 10 ARM-based PCs with a Microsoft SQ1, Microsoft SQ2, Qualcomm Snapdragon 8cx, or Qualcomm Snapdragon 850 processor
- Windows 10 Pro or Enterprise, build 19559 or newer
- Hyper-V enabled (instructions)

Instructions

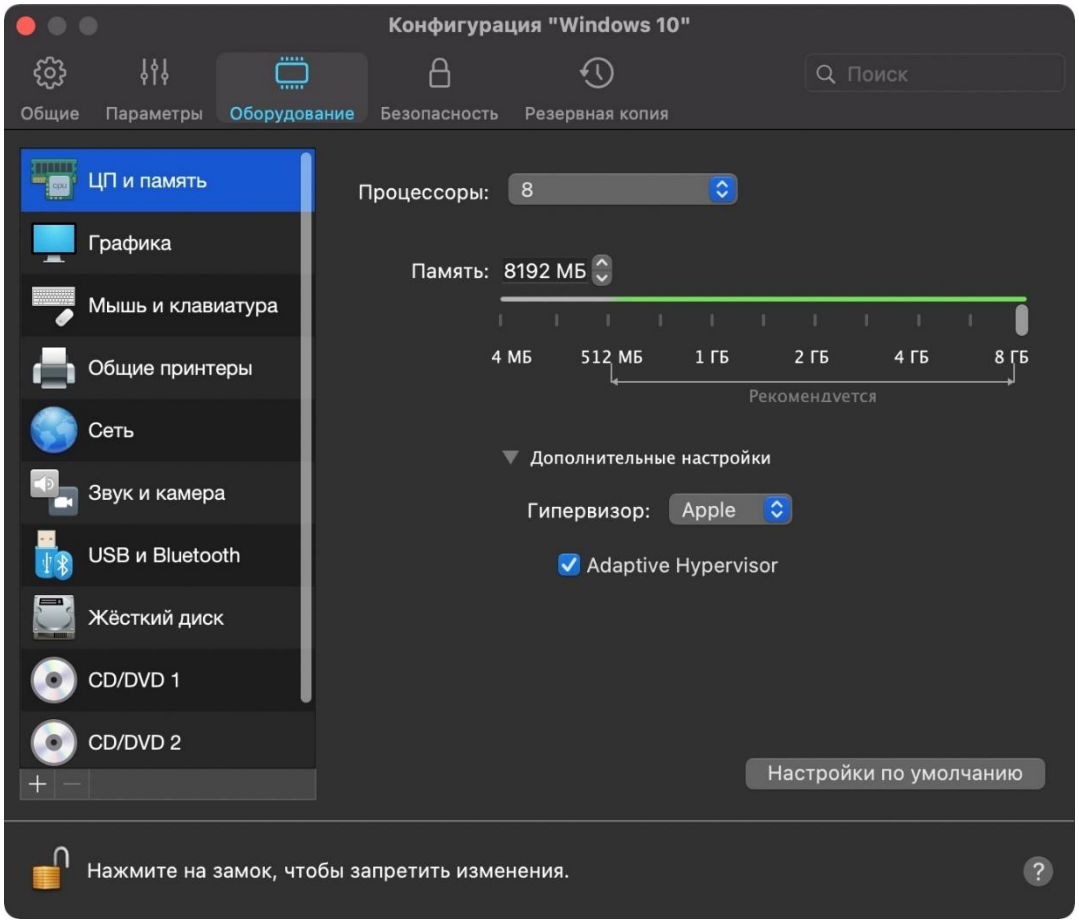
1. Download VHDX file from this page
2. Create a virtual machine in Hyper-V, using the downloaded VHDX as an existing virtual hard disk

Visit Windows 10 on ARM developer center at http://aka.ms/win10onarm for more details and documentation.

Windows 10 Client ARM64 Insider Preview - Build 21286

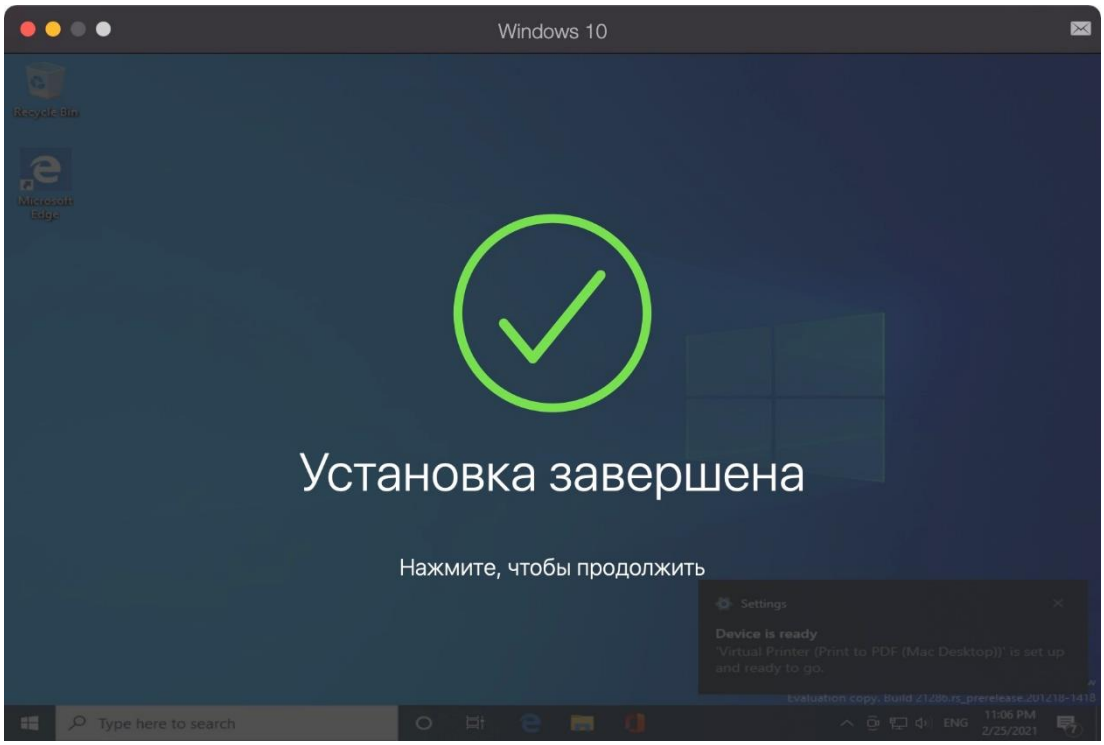
Скриншот 214. Требования образа виртуальной машины Windows 10 on ARM.

Далее регистрируете себе аккаунт Microsoft и вступаете в программу Windows Insider Preview, после чего Вам становится доступна страница загрузки образа виртуальной машины Windows 10 on ARM Insider Preview.



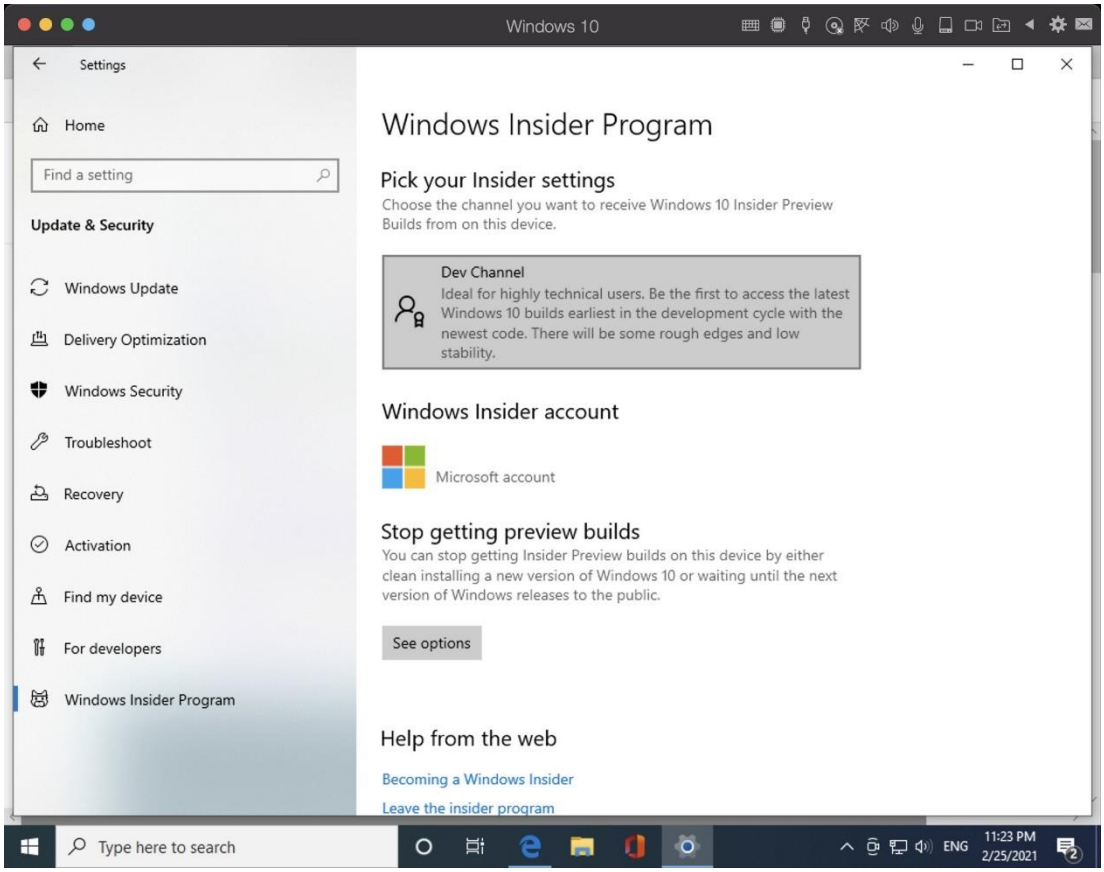
Скриншот 215. Максимально доступный объём оперативной памяти для виртуальной машины Parallels.

По настройкам: виртуальной машине нельзя выделить более 8 ГБ объединённой памяти, так что для решения задач, требующих более 8 ГБ оперативной памяти в Windows, Macbook Pro с Apple M1 сразу отпадает.



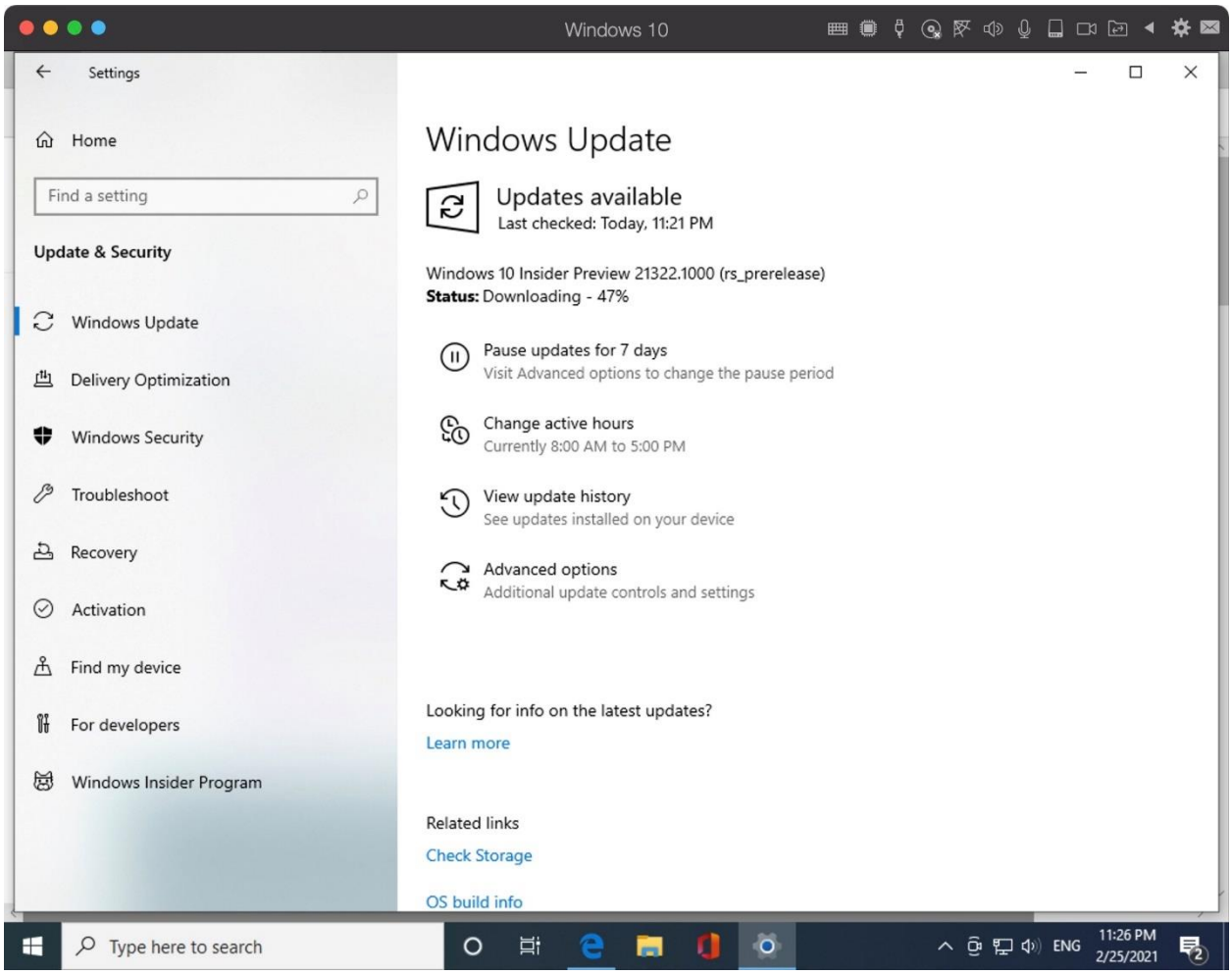
Скриншот 216. Установка образа виртуальной машины Windows 10 on ARM в Parallels Desktop.

После того, как мы установили Windows, его нужно ещё обновить.



Скриншот 217. Переход на Dev канал обновлений Windows 10 Insider Preview.

Чтобы обновить ОС, нужно сперва перейти на канал обновлений для разработчиков, иначе останетесь со старой версией после установки, а эмуляция x86_64 приложений в Windows появилась лишь в свежей версии.

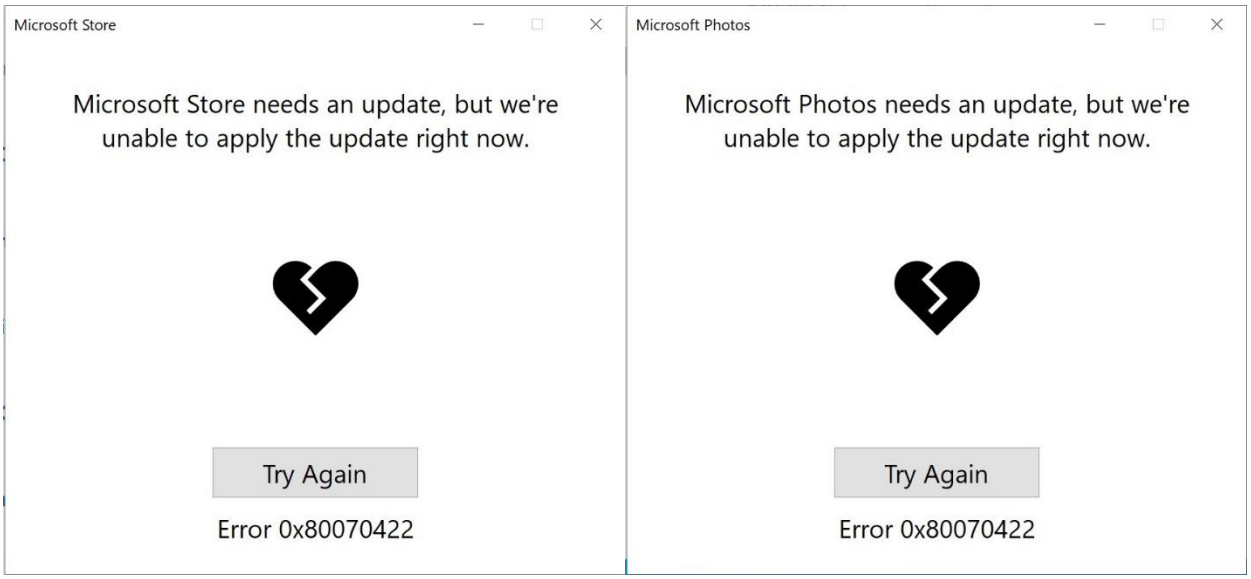


Скриншот 218. Обновление Windows 10 Insider Preview.



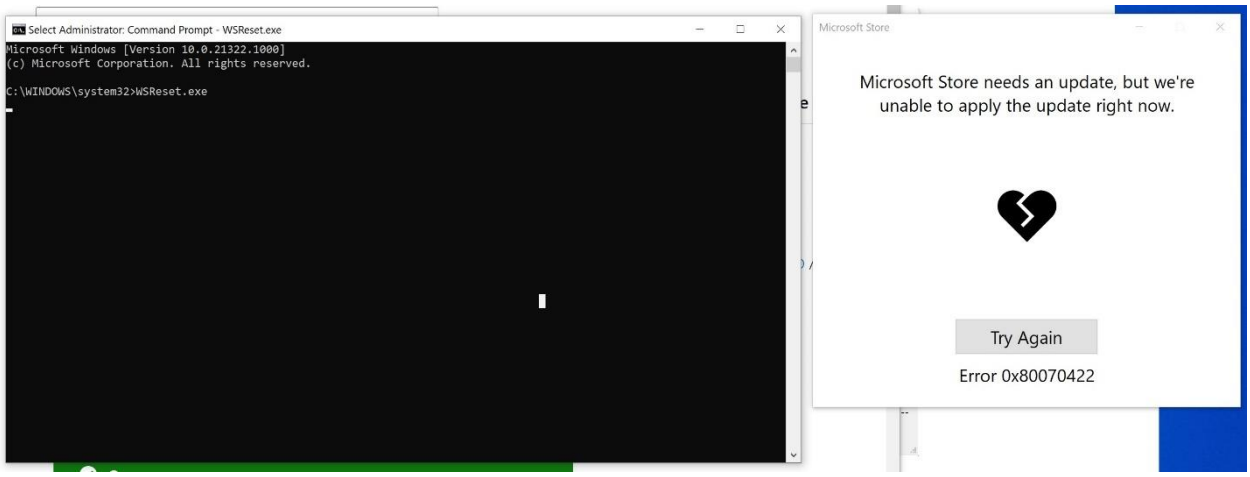
Скриншот 219. Версия Windows 10 Insider Preview до и после установки обновлений.

Осталось установить обновления приложений из Microsoft Store.



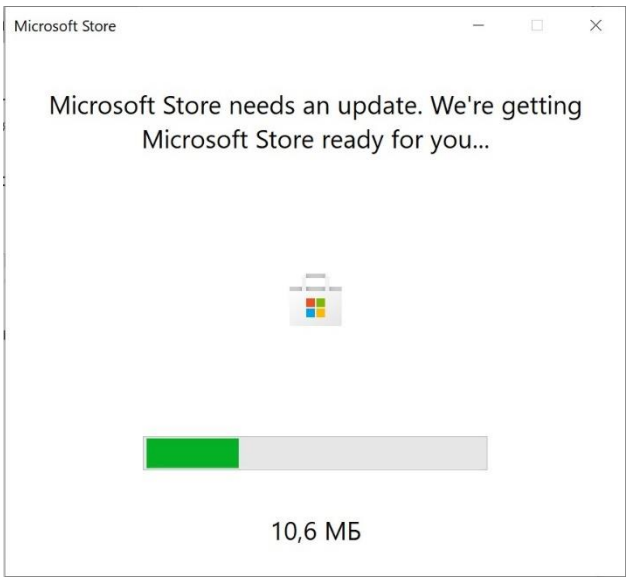
Скриншот 220. Ошибка при запуске предустановленных UWP приложений.

Странно. Может, попробуем при помощи WSReset оживить MS Store?



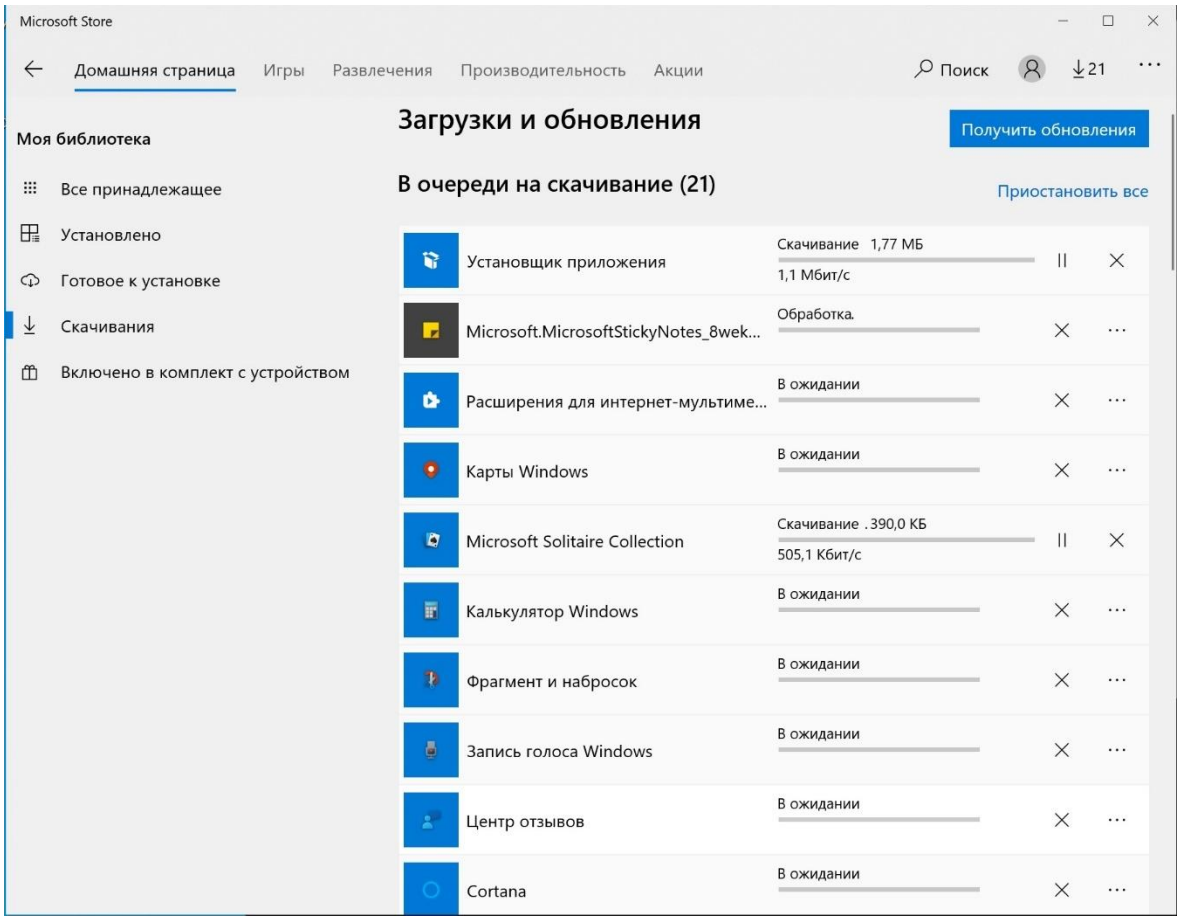
Скриншот 221. Попытка оживить Microsoft Store командой WSReset.exe.

Какое досадное недоразумение. Я уже, было, хотел забить на всё, но сперва мне пришло в голову попробовать перезагрузить ВМ. Сработало.



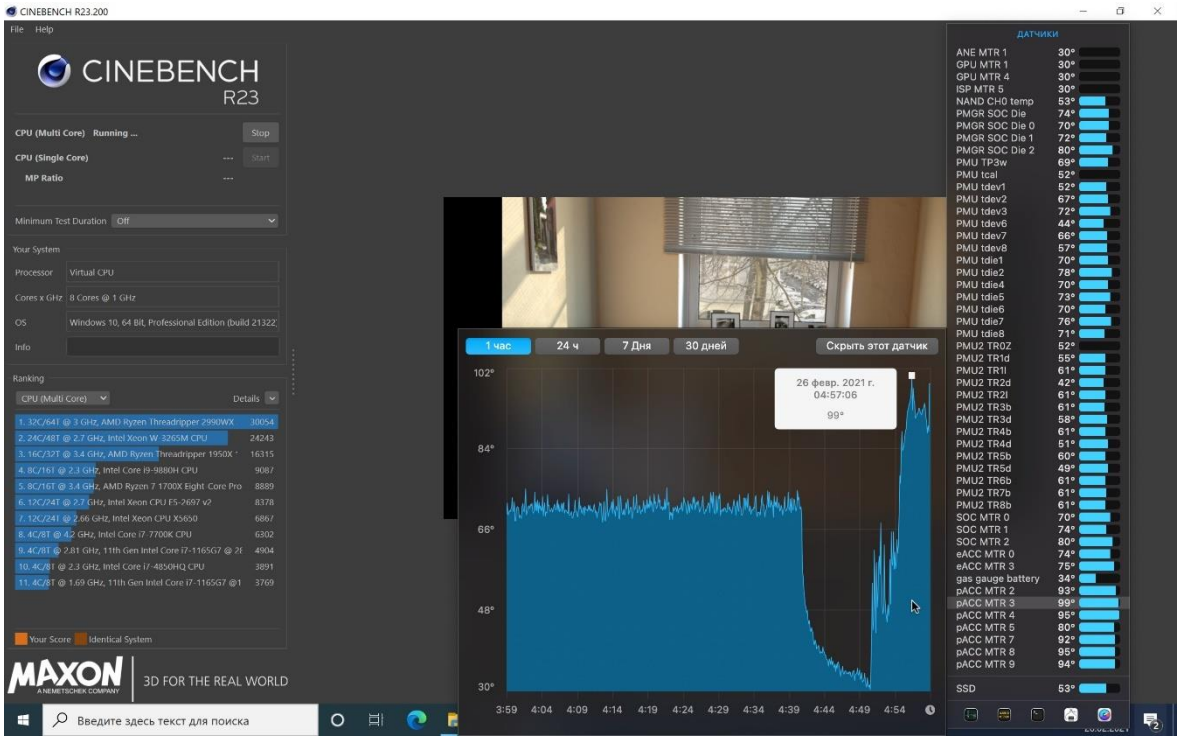
Скриншот 222. Запуск Microsoft Store после перезагрузки виртуальной машины.

Ну, ладно, с кем разок косяка не бывает, верно?



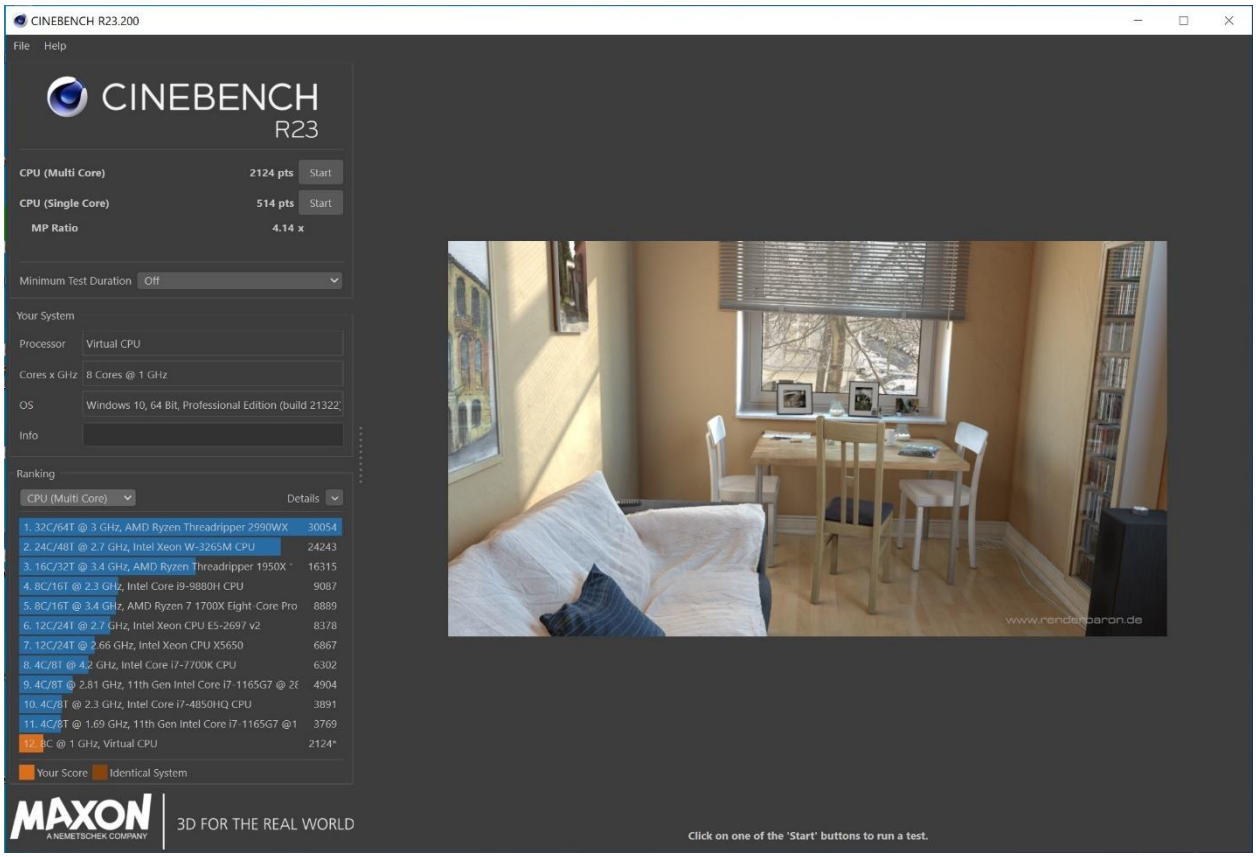
Скриншот 223. Обновление предустановленных приложений через Microsoft Store.

Сразу скажу, что обновления были установлены все, какие были доступны, т.е. я тестировал всё с самой свежей на тот момент версией ПО.



Скриншот 224. Нагрев CPU во время теста Cinebench R23.

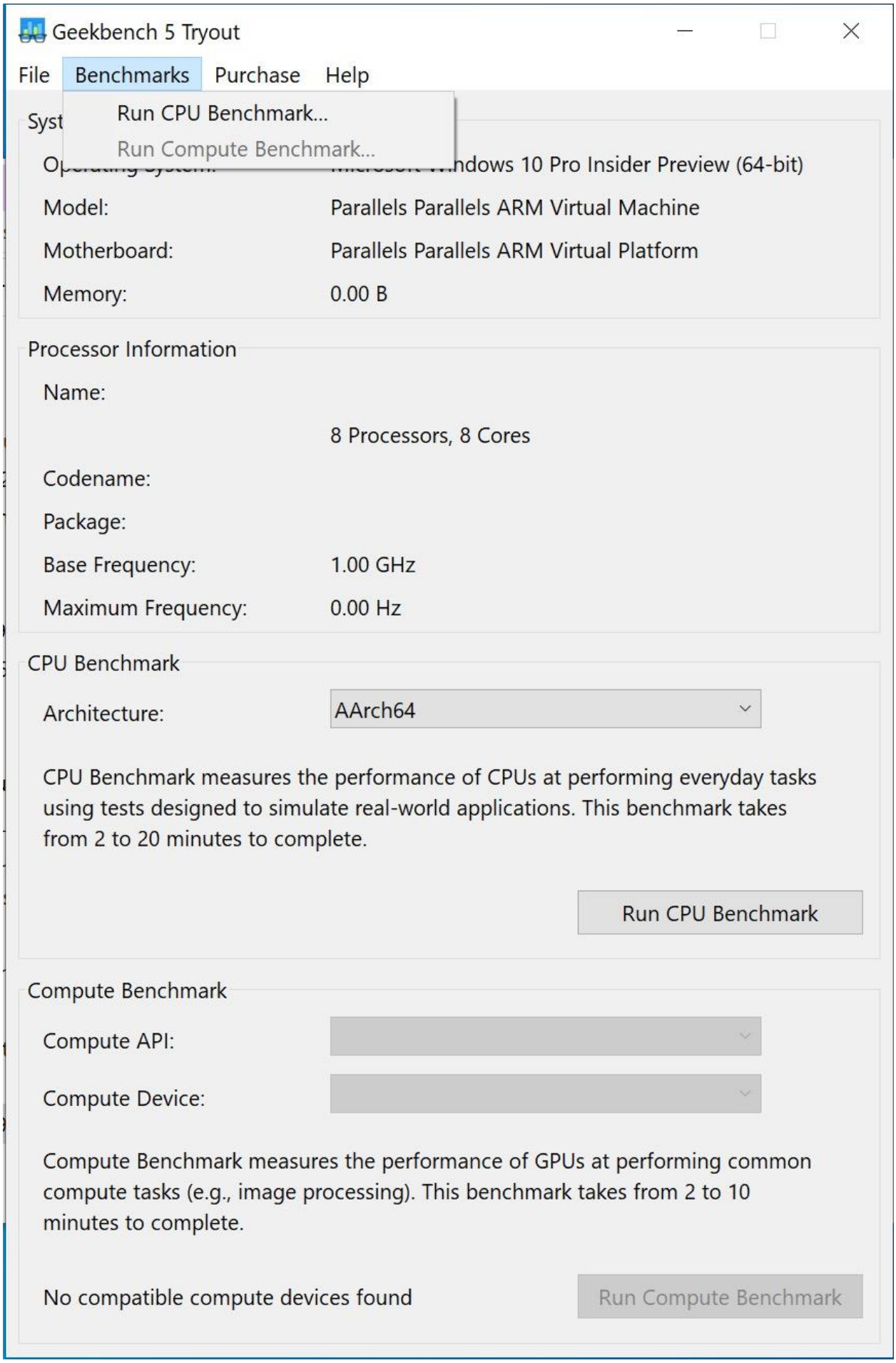
Я решил провести бенчмарки, чтобы определить, насколько снижается производительность при использовании виртуальной машины Windows. Грелся в тестах MBP 13 с Apple M1 в Parallels намного сильнее, чем в macOS.



Скриншот 225. Результат Macbook Pro 13 2020 (Apple M1, 20W) в тесте Cinebench R23 в Windows 10 (Parallels).

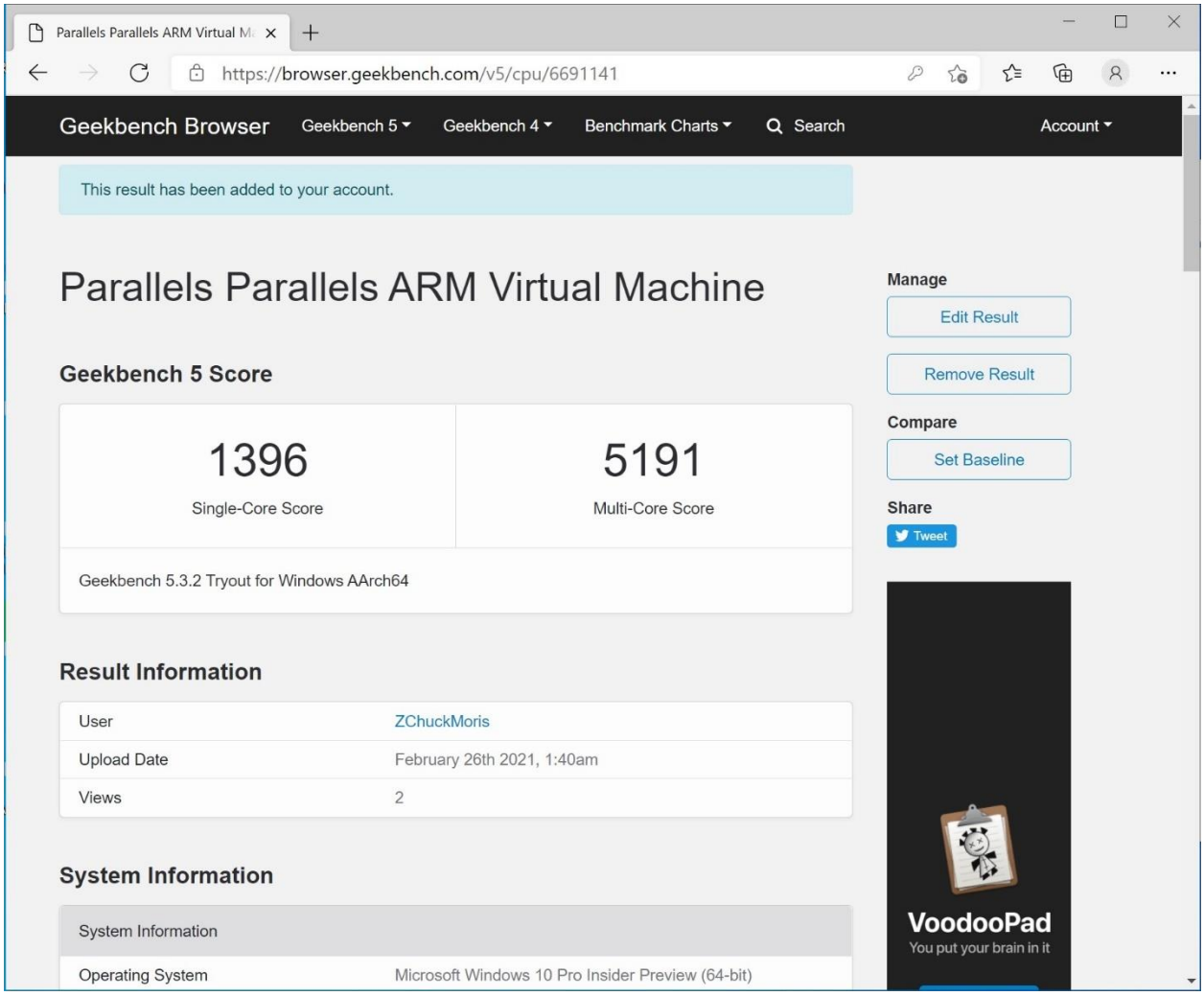
И дело не только в нагреве. Сравните эти показатели с теми, что у нас были в Cinebench R23 ранее. 7807 баллов набирал Macbook Pro 13 с Apple M1 в тесте Cinebench R23 без Rosetta 2, 5184 балла он набрал с Rosetta 2 и вот сейчас он набрал 2124 балла с виртуализацией Parallels Desktop и эмуляцией x86_64 платформы в Windows 10. В сравнении с нативным исполнением программ на macOS разница 3,68 раза, а в сравнении с Rosetta 2 на macOS – 2,44 раза. По одноядерной производительности там просадка 2,92 раза и 1,92 раза соответственно.

Понимаете ли, дело в том, что на Windows нативного ПО под ARM практически нет. Т.е. если в macOS расклад примерно 50/50, то в Windows расклад 1/99, и нет на данный момент никаких причин считать, что в ближайшее время нативного ПО под Windows для ARM64 станет больше.



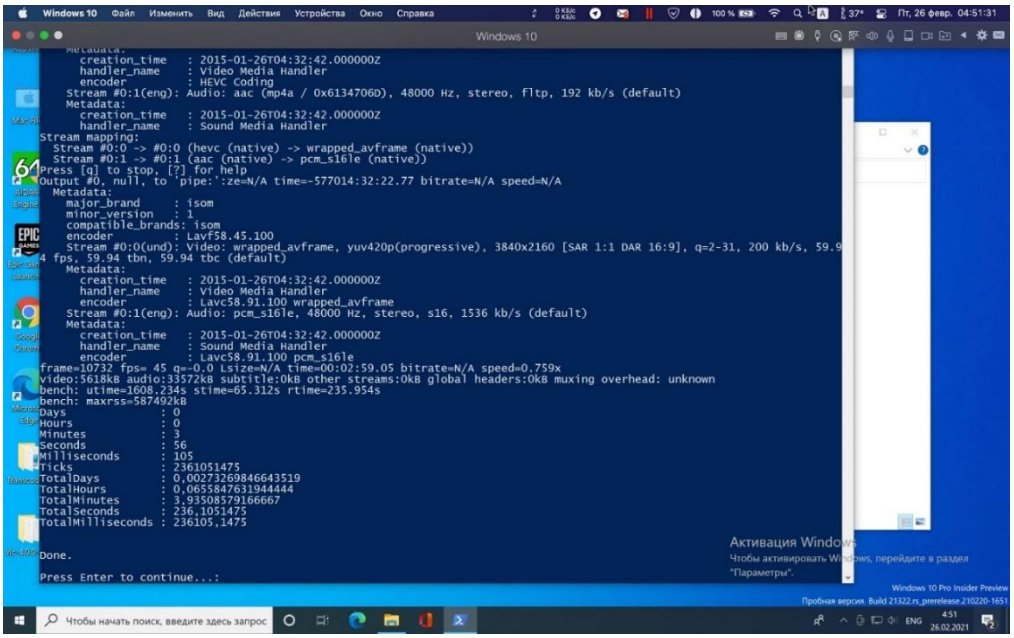
Скриншот 226. Невозможность проведения Compute теста в Geekbench 5.

В Geekbench 5 у нас нет выбора API: OpenCL или Vulkan. Мы просто никак не можем задействовать видеокарту для проведения вычислений.



Скриншот 227. Результат теста в Geekbench 5 в Parallels. Тест нативно в aarch64, без эмуляции x86_64.

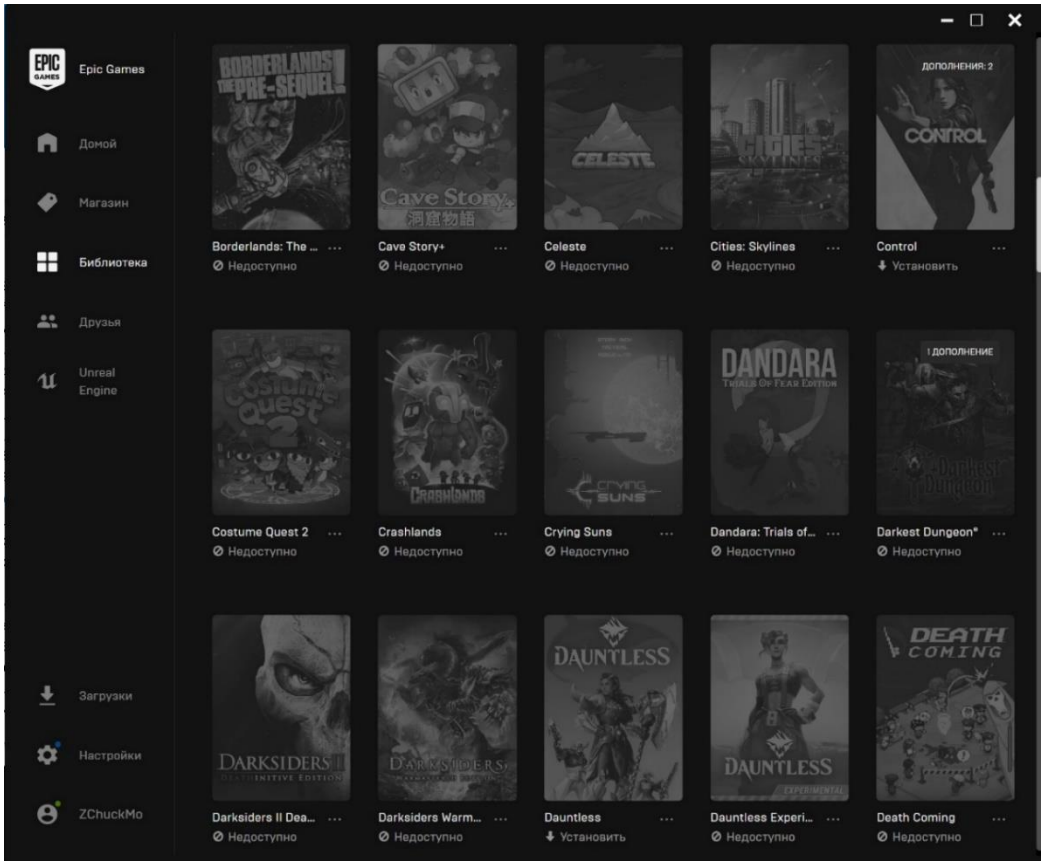
В Geekbench 5 тест проводился в нативном режиме aarch64 (ARM64) без эмуляции платформы x86_64, и итоговый результат составил 1396 баллов в Single Core и 5191 баллов в Multi Core. Для сравнения, в macOS в нативном режиме (aarch64/ARM64) результат составил 1746 баллов в Single Core и 7669 баллов в Multi Core, а в режиме Intel (x86_64) результат составил 1340 и 6051 баллов соответственно. Выходит, что в Single Core результат с просадкой в производительности из-за виртуализации выходит сопоставимым с таковым в macOS при использовании Rosetta 2 и на 20% ниже, чем при нативном исполнении в macOS. В Multi Core же результат на 14,2% ниже, чем в macOS при использовании Rosetta 2, и на 32,3% ниже, чем в нативном режиме в macOS. Просадка по производительности удручающая.



Скриншот 228. Скорость декодирования 4K 60 FPS H.265 8-bit на CPU в ffmpeg в Parallels.

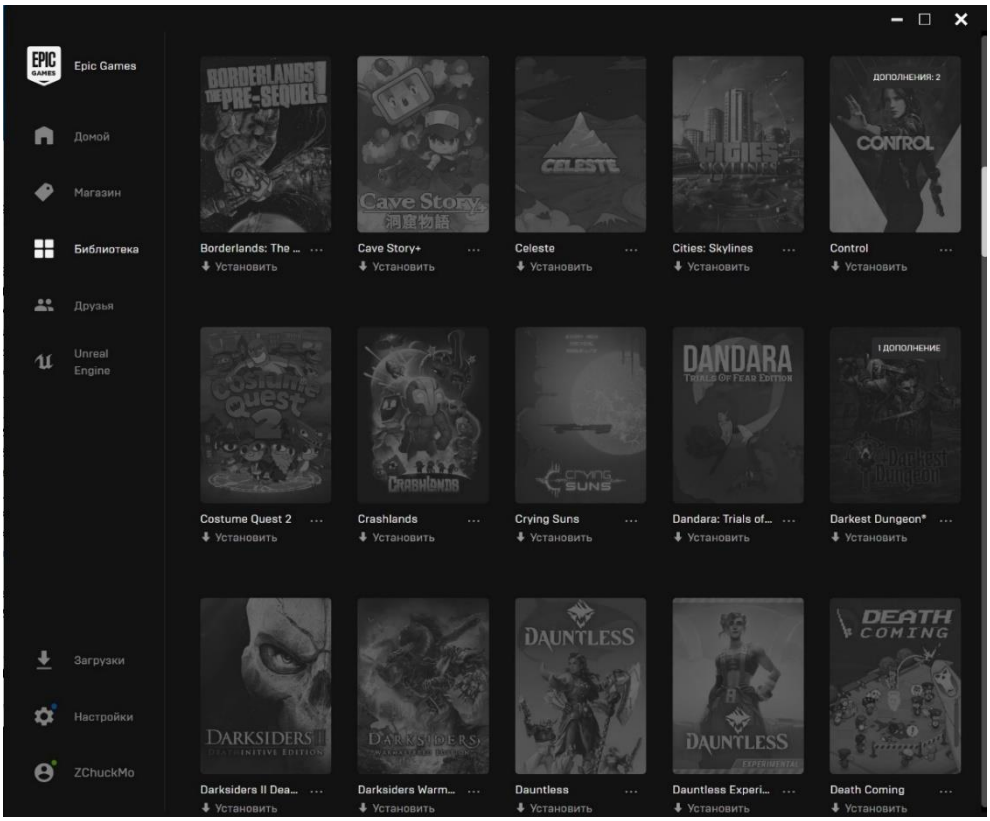
Напоследок я проверил, с какой скоростью ffmpeg декодирует 4K 60 FPS H.265 8-bit видео на CPU. Результат в 1,39 раза ниже, чем нативно в macOS, и в 2,26 раза чем с Rosetta 2 (да, если помните, с ней было быстрее).

Я также собрал данные по скорости перекодирования на CPU: 3782 секунды, и 4784 секунды с опцией -crf 0, что в 2.7-2.75 раза дольше, чем в macOS без Rosetta 2, и в 1.53-1.92 раза дольше, чем в macOS без Rosetta 2.



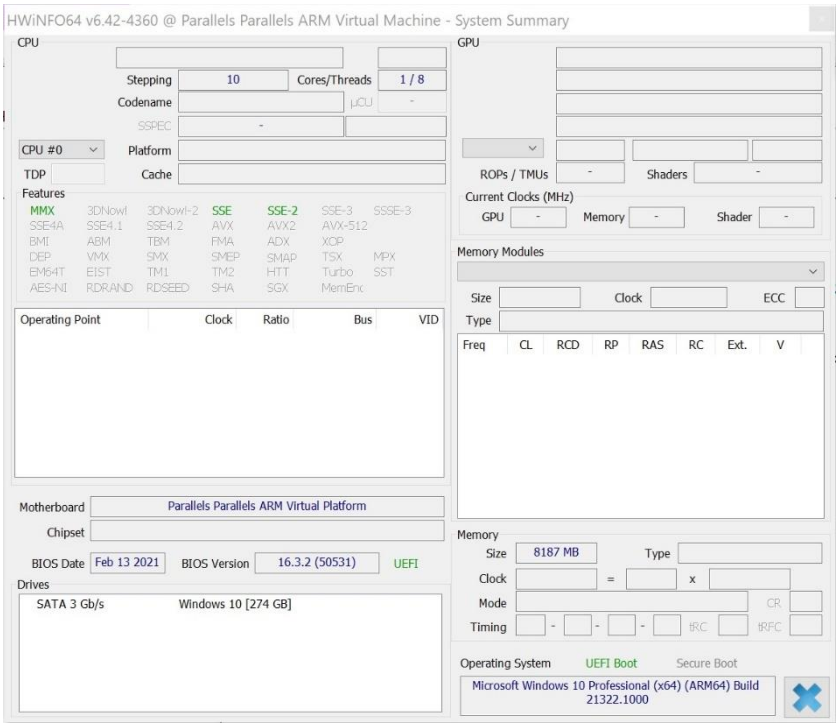
Скриншот 229. Отсутствие возможности установки игр в Epic Games Store в Windows в Parallels.

И вот опять ничего не доступно (кроме Control). Попробуем исправить.



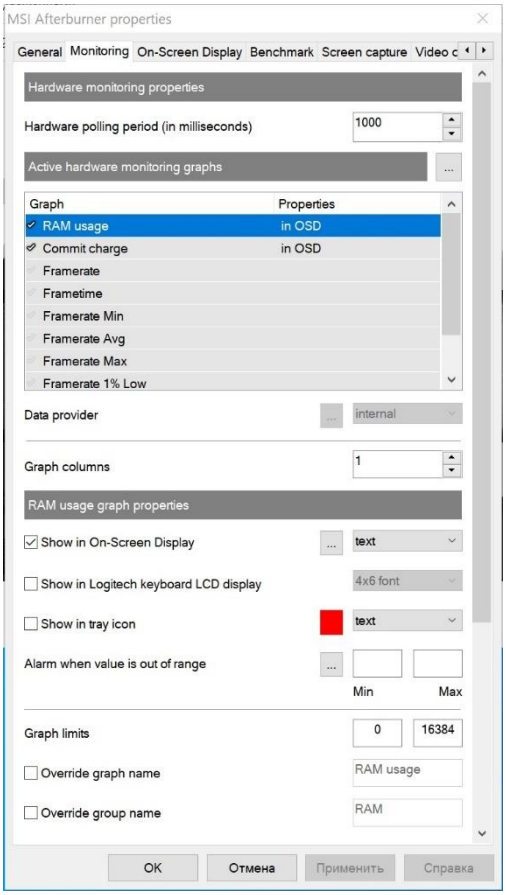
Скриншот 230. Наличие возможности установить игры в Epic Games Store после очередной перезагрузки.

Что я сделал? Всего лишь в очередной раз перезагрузил виртуальную машину и наконец-то, хотя бы, появилась возможность установить игры из Epic Games Store. Сюда бы подошёл мем «а вы пробовали выключить и включить?». Я не знаю, Вы хотите так каждый раз перезагружать ВМ?



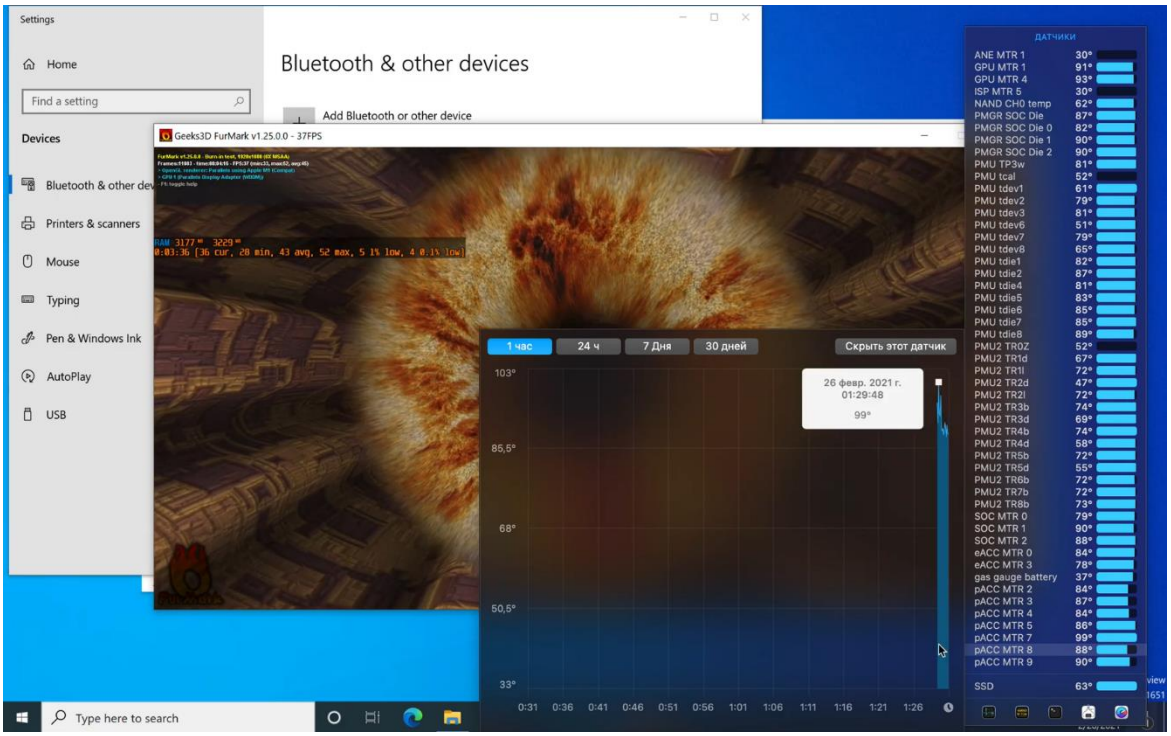
Скриншот 231. Отсутствие информации в HWinfo.

Я решил посмотреть информацию по оборудованию в HWinfo, но тут пусто. С другой стороны, я и без того в macOS о железе узнал всё, что хотел.



Скриншот 232. Доступная для мониторинга информация в MSI Afterburner.

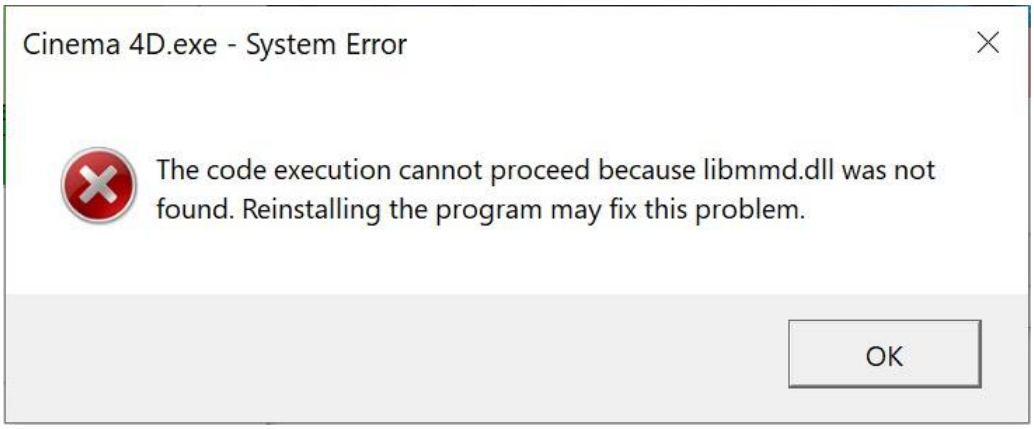
Ну, в MSI Afterburner у нас информации не особо то и много, но нам хватит и информации о FPS: средний FPS. меньший 1% FPS и макс. FPS.



Видео 10. Мониторинг MSI Afterburner в Furmark.

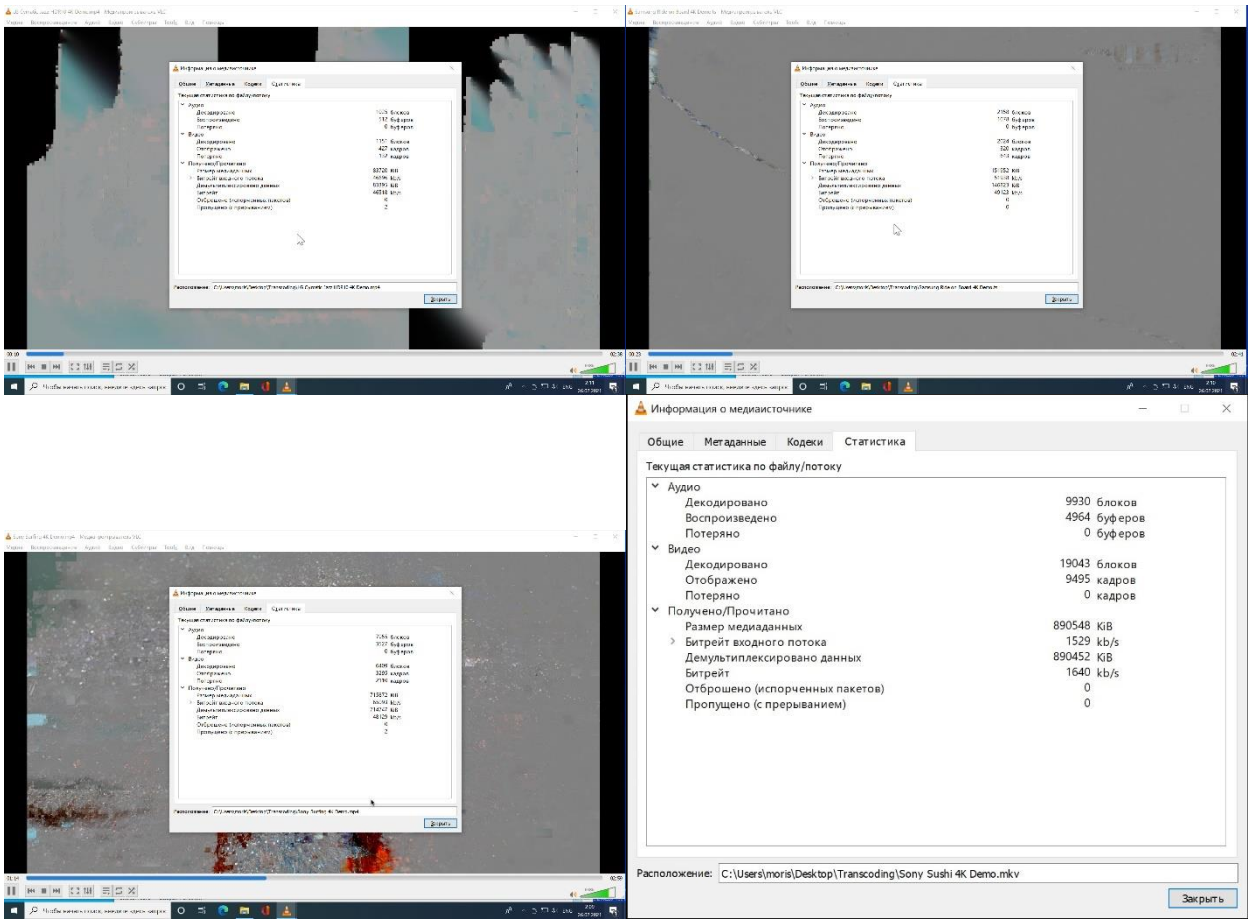
Вот как Вы считаете, информации из MSI Afterburner можно верить в случае работы из-под виртуальной машины Parallels? Это похоже на 43 FPS в среднем? Разве можно полагаться на MSI Afterburner на Mac на Apple M1?

С ноутбуками на родном обычном x86_64 Windows 10 я такого не наблюдал ни разу. И тут началось.....



Скриншот 233. Ошибка при запуске Cinema 4D на Windows 10 on ARM в Parallels.

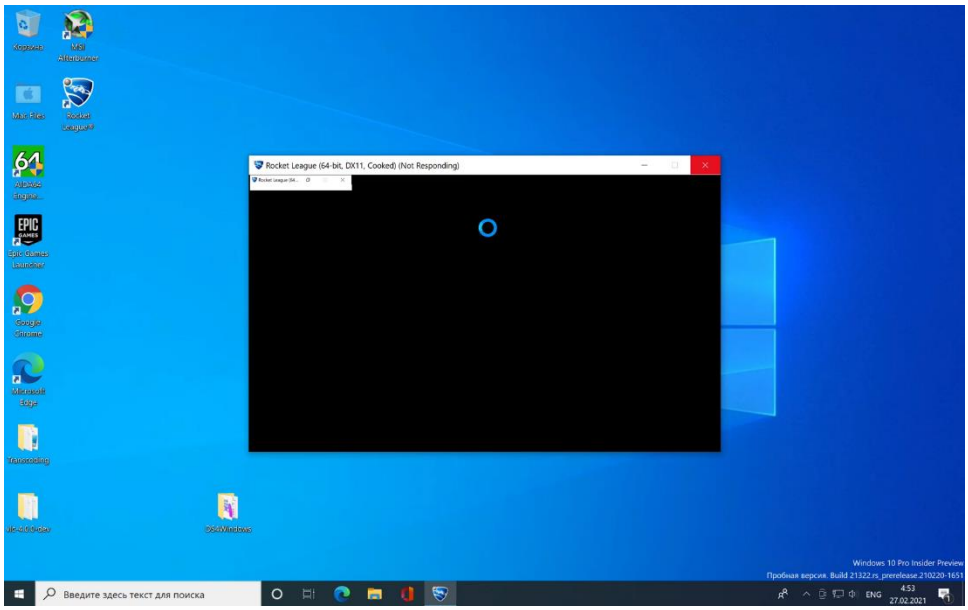
Программы просто одна за другой отказались запускаться. Да, зачем нужен Cinema 4D в Windows, когда он есть в macOS, но я-то не знаю, какой Вам софт на винде нужен, так что пробую что есть, что грузит железо.



Скриншот 234. Проблема с проигрыванием 4K 60 FPS H.265 (HEVC) видео в VLC. Проигрывается только H.264 8-bit.

Я установил 4.0 dev (developer) версию VLC, т.к. на сайте VLC было написано, что эта версия поддерживает Windows 10 on ARM, но Вы сами всё видите. Нормально воспроизвёлся только ролик в H.264, никакого Вам H.265.

И ОМГ, Вы даже не представляете, насколько Chrome с эмуляцией x86_64 лагает внутри виртуализированной Windows 10 on ARM.



Скриншот 235. Невозможность запуска Rocket League в Windows 10 on ARM в Parallels.

Это была точка невозврата. Я утратил веру в запуск Windows игр на Mac с Soc Apple M1. Даже Rocket League не запускается! Кому тогда вообще нужна Windows? Я запускал превью версию программы (VLC) для ARM в превью версии Операционной Системы (Windows 10) для ARM в превью версии Виртуальной Машины (Parallels Desktop) для ARM в непонятно какой, с точки зрения стабильности, версии Операционной Системы (macOS) для ARM. И после всего я даже Rocket League не могу запустить. Как так-то?

Ладно, если серьёзно, про Windows приложения просто забудьте, если хотите себе Mac на M1. Если нужен Windows, оставьте себе запасной компьютер и пользуйтесь им время от времени, в т.ч. и по удалёнке. И да, всё, что говорил Роман Ястреб по части Windows на M1 - абсолютная правда.

UPDATE: пока я дописывал эту статью вышла стабильная версия (16.5) Parallels Desktop для M1, но я её уже не протестирую, т.к. отформатировал Mac и сдал его обратно. Не думаю, что ситуация со стабильной версией Parallels Desktop будет сильно отличаться, но я этого уже не могу проверить.

10. Тестирование игр. Методика и результаты.

10.1. Методика тестирования.

По тестам GPU у меня сложилось стойкое впечатление, что по своей голой производительности интегрированный видеоускоритель в Apple M1 не уступает видеокарте GTX 1050 Max-Q в моём ноутбуке Xiaomi. Но нюанс в том, что они сопоставимы только без использования Rosetta 2. Если транслировать код из x86_64 в ARM64 при помощи Rosetta 2, интегрированный видеоадаптер в Apple M1 будет значительно отставать от моего GTX 1050 Max-Q, т.к. Rosetta 2, всё-таки, не со 100% КПД транслирует код с одной архитектуры в другую. Сгенерированный по итогу код менее эффективный, и потому скорость взаимодействия CPU и GPU снижается.

Более того, чаще всего, игры в первую очередь оптимизируют под ОС Windows с её API DirectX, а адаптация игр под Mac с Metal и Linux с Vulkan, как правило, - дело десятое. Хоть разработчики того же Tomb Raider и пишут о наилучшей адаптации под все платформы, но им просто не может хватить ресурсов для идеальной оптимизации проекта под все ОС и аппараты сразу.

На YouTube хватает сравнений одних и тех же игр на одних и тех же компьютерах Mac на macOS и на Windows, и на Windows там, как правило, FPS значительно выше. Примеры таких сравнений на компьютерах Mac: Macbook Pro 15 2018 (i9 8950HK, 560X, 32 GB RAM) от MrMacRight, iMac 27” 5K 2019 (i9 9900K, Vega 48 8 GB, 16 GB RAM) от MrMacRight, Macbook Pro 16 2019 (i9, 5500M, 16 GB RAM) от Mikes Tech и Macbook Pro 16 2019 (i9, 5500M, 16 GB RAM) от Романа Ястреба

Также можете ознакомиться со статьёй на портале barefeats, где сравнивается FPS в Rise of the Tomb Raider на macOS (API Metal) и Windows (API DirectX 11 и API DirectX 12) на iMac Pro 2017 (Xeon W-2140B 8-Core 3.2 GHz, 32 GB ECC RAM, Radeon Pro Vega 64 16 GB).

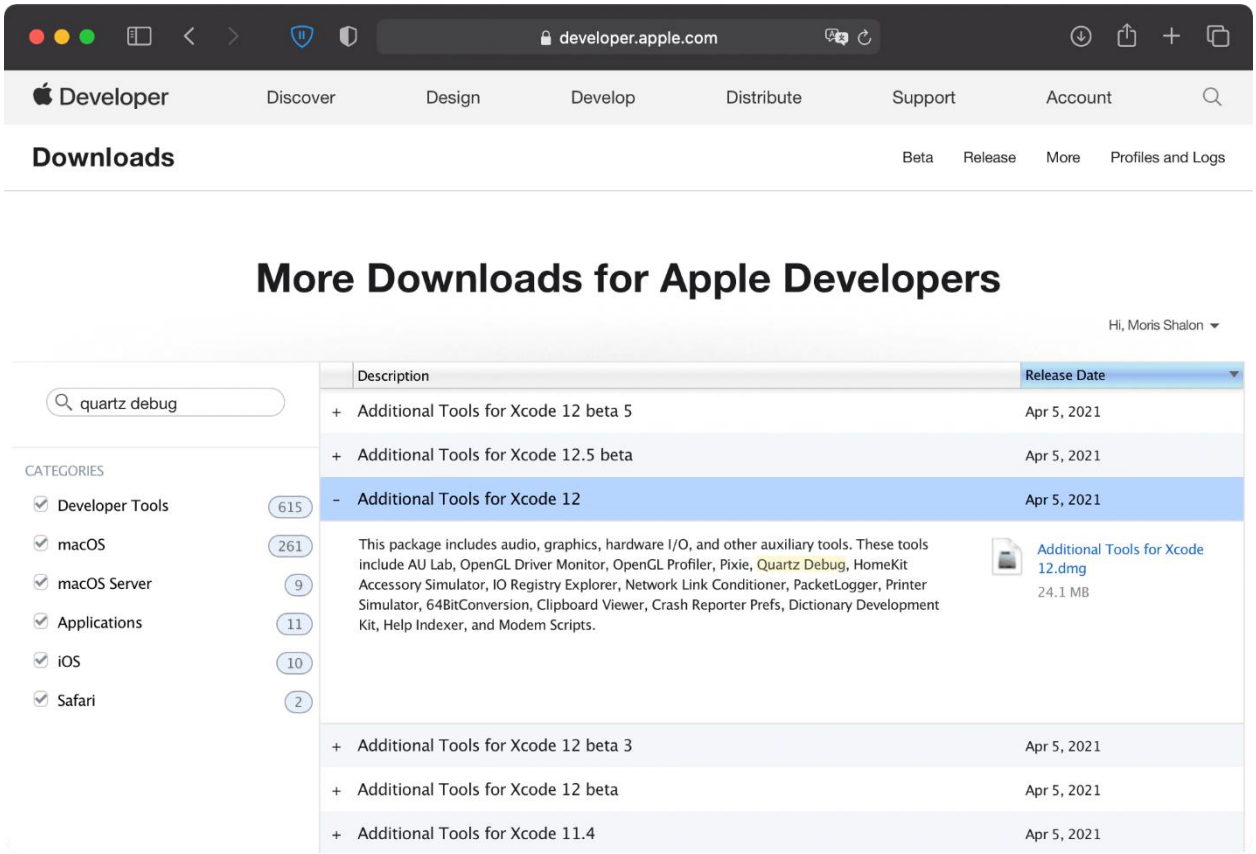
Во всех сравнениях FPS на Windows выше в случае с серией игр Tomb Raider, поэтому имейте сразу в виду, что факторов, по которым у нас FPS будет ниже на M1 сразу 2: Rosetta 2 и недостаточная оптимизация под API

Metal. Хотя, возможно, API Metal априори не столь оптимален для разработки игр, сколь API DirectX, но это уже тема для отдельной статьи.

Можно ли провести абсолютно релевантное сравнение? Нет, нельзя. Под macOS на сегодняшний день нет ни одной AAA игры с высокой нагрузкой на графический ускоритель, работающей при этом нативно на Apple Silicon. Т.е. даже если я раздобуду сопоставимый по производительности GPU Macbook предыдущих поколений, сравнение будет уже не релевантно из-за Rosetta 2.

Мы можем лишь примерно сопоставить GPU в Macbook Pro 13 с Apple M1 с таковым в любом другом аппарате, сделав скидку на Rosetta 2 и на разницу в адаптации игр под API Metal и API DirectX 11 или DirectX 12, если сравниваем результат не только с Mac, но и с Windows аппаратами.

В macOS нет полноценного аналога MSI Afterburner с его оверлеем, и это сулит нам некого рода проблемы, но есть хоть какая-то альтернатива.

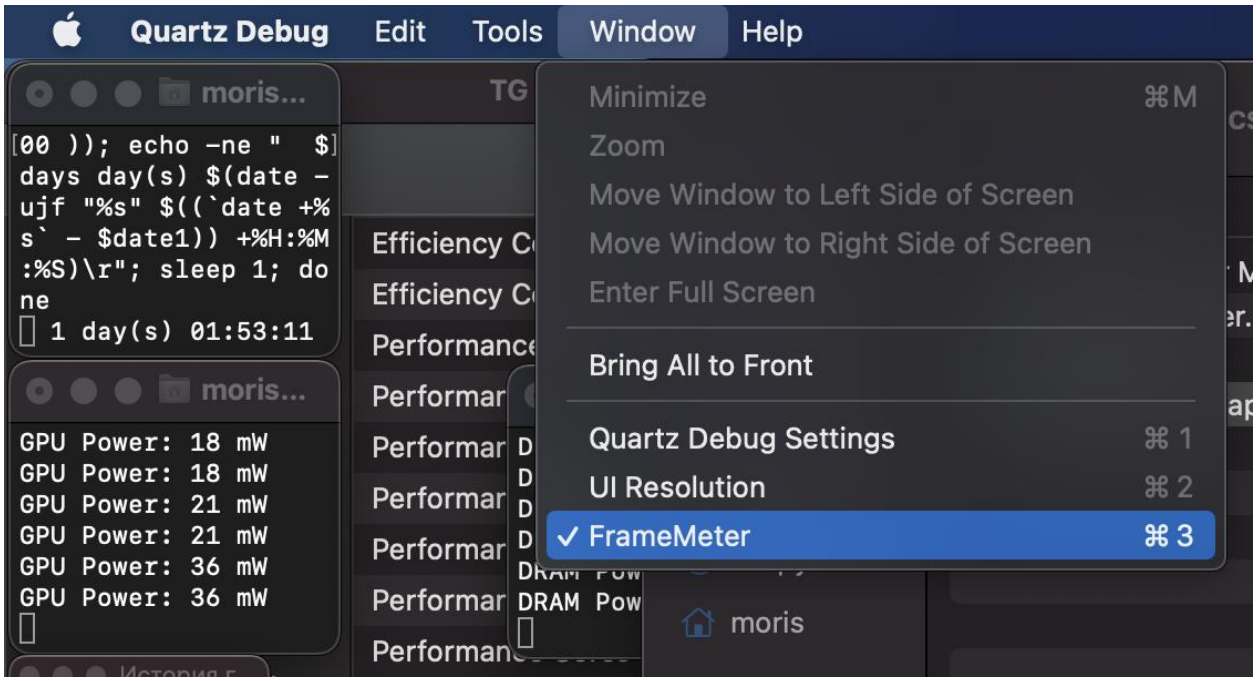


Скриншот 236. Дополнительные инструменты для XCode 12, включающие пакет Quartz Debug.

Переходим на раздел загрузок портала Apple для разработчиков, авторизуемся своим Apple ID (сертификат разработчика с ежегодной оплатой в 100$ не обязателен), вводим в поиске Quartz Debug, жмём Enter и

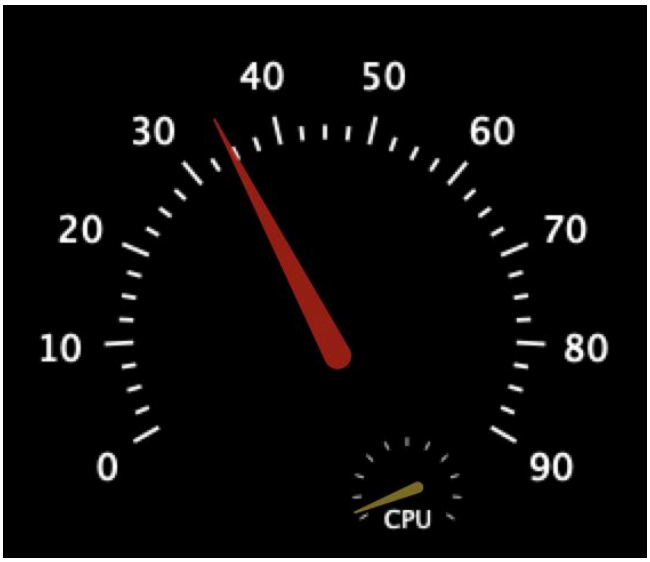
загружаем себе Additional Tools for XCode (последнюю доступную не-бета версию, на данный момент – утилиты для версии для XCode 12).

Из загруженного .dmg образа извлекаем все файлы в любую новую папку на рабочем столе (можете так её и назвать: «XCode-Additional-Tools»), в ней в папке Graphics запускаем приложение Quartz Debug.app.



Скриншот 237. Активация FrameMeter в Quartz Debug.

Далее в menu bar нажимаем на Window и выбираем FrameMeter.



Скриншот 238. Мониторинг FPS через Quartz Debug.

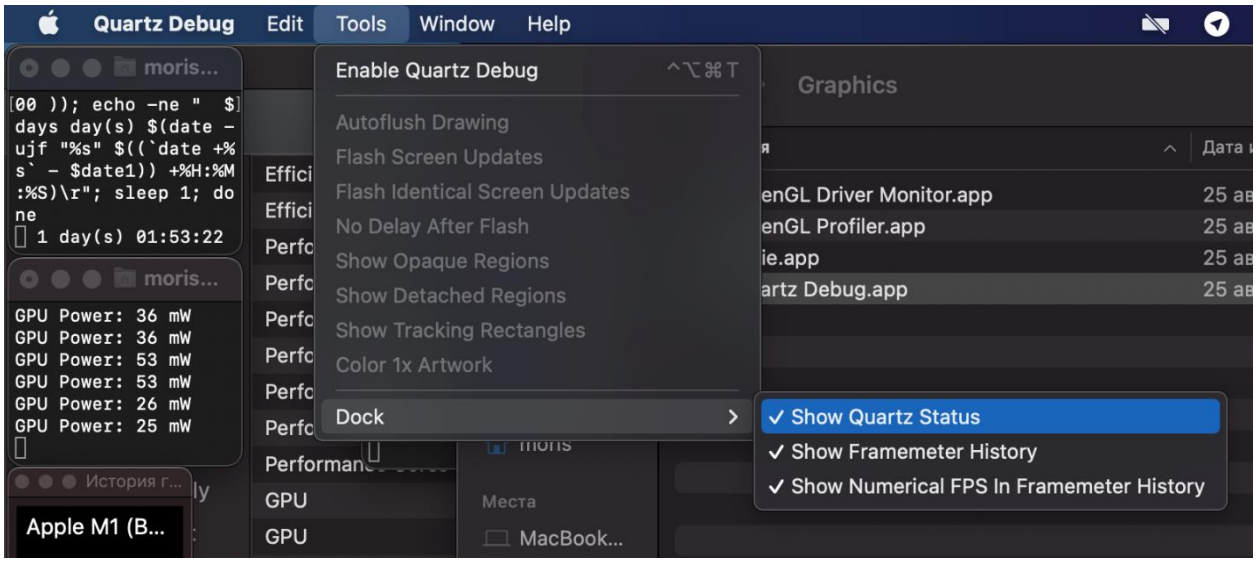
После этого мы видим окно с мониторингом FPS, на вид как автомобильный тахометр. И тут есть очень важный нюанс: Quartz Debug отслеживает не число кадров в секунду в приложении, задействующем API Metal, а общее число кадров, показанных за прошедшую секунду на экране.

В чём тут проблема? Дело в том, что у самого «тахометра» FPS – 25-30.



Скриншот 239. Число кадров в секунду, замеренных Quartz Debug.

В отличие от MSI Afterburner в Windows, который почти не оказывает влияния на результат, здесь же ниже 25 FPS не будет из-за «тахометра» FPS.



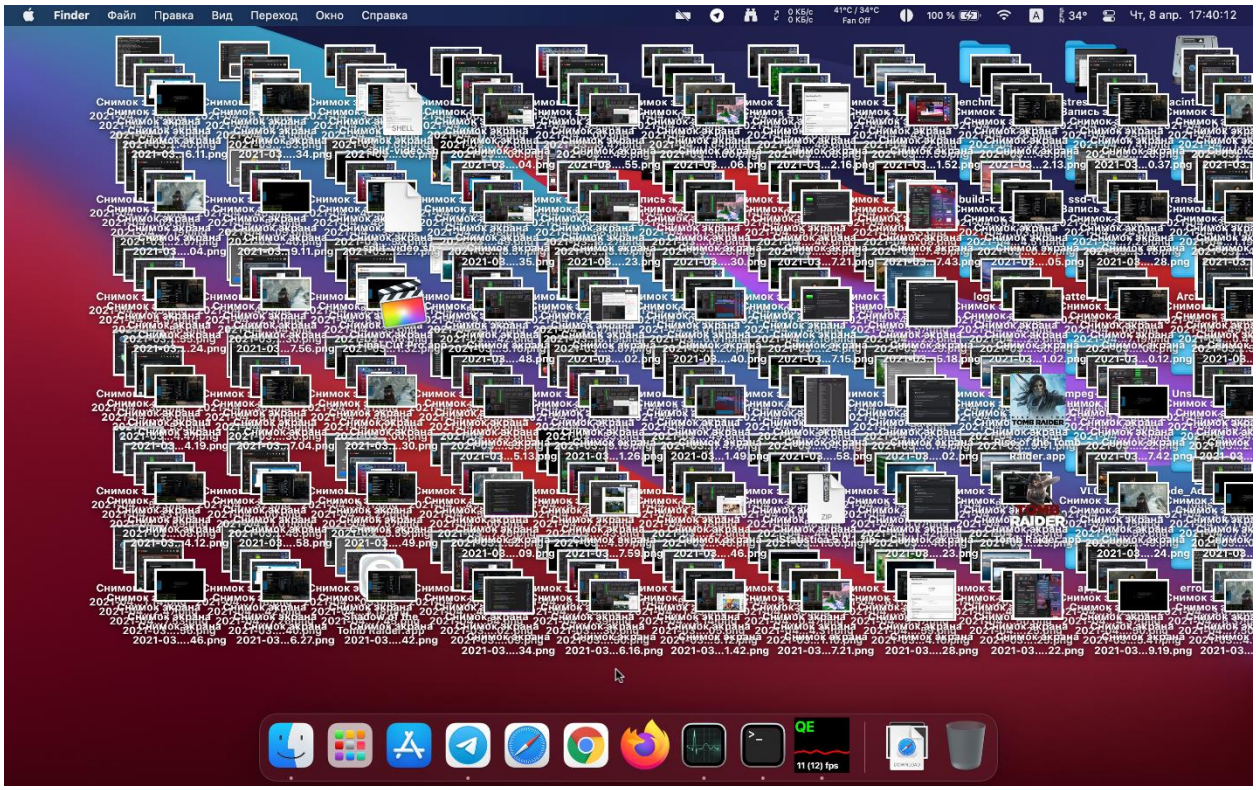
Скриншот 240. Активация отображения информации из Quartz Debug в Dock.

Есть ещё другой вариант: мы можем активировать отображение информации из Quartz Debug в Dock на иконке самого приложения.



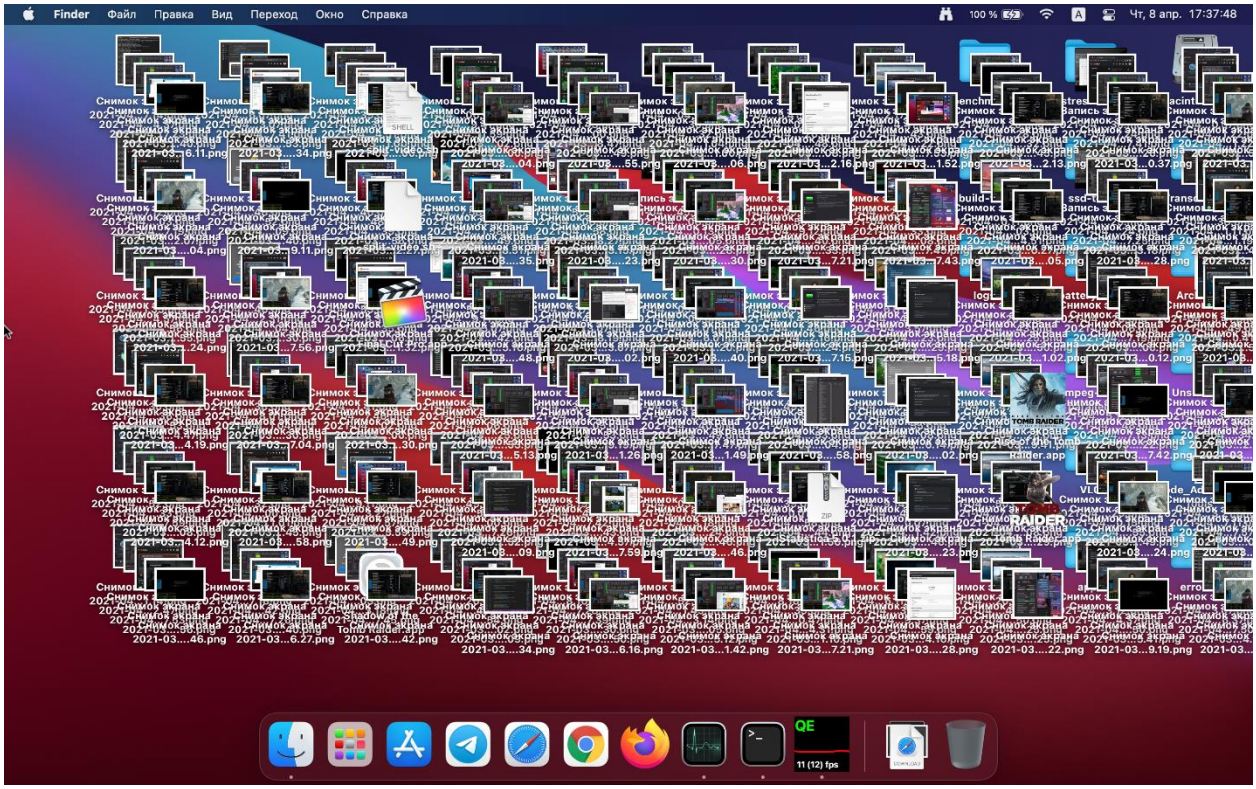
Скриншот 241. Мониторинг FPS через Quartz Debug в Dock.

Когда на экране приложения реже не обновляют свой контент и нет тахометра, FPS, замеренный в Quartz Debug в Dock, составляет всего 14-16 (16 в скобках это максимум, который был замерен за последние 10 секунд).



Скриншот 242. Число подсчитанных FPS в Dock без приложений на экране со сторонними приложениями в тени bar.

Если же убрать с экрана всё лишнее, будет замерено не 14-16 обновлений контента на экране за секунду, а 11-12.



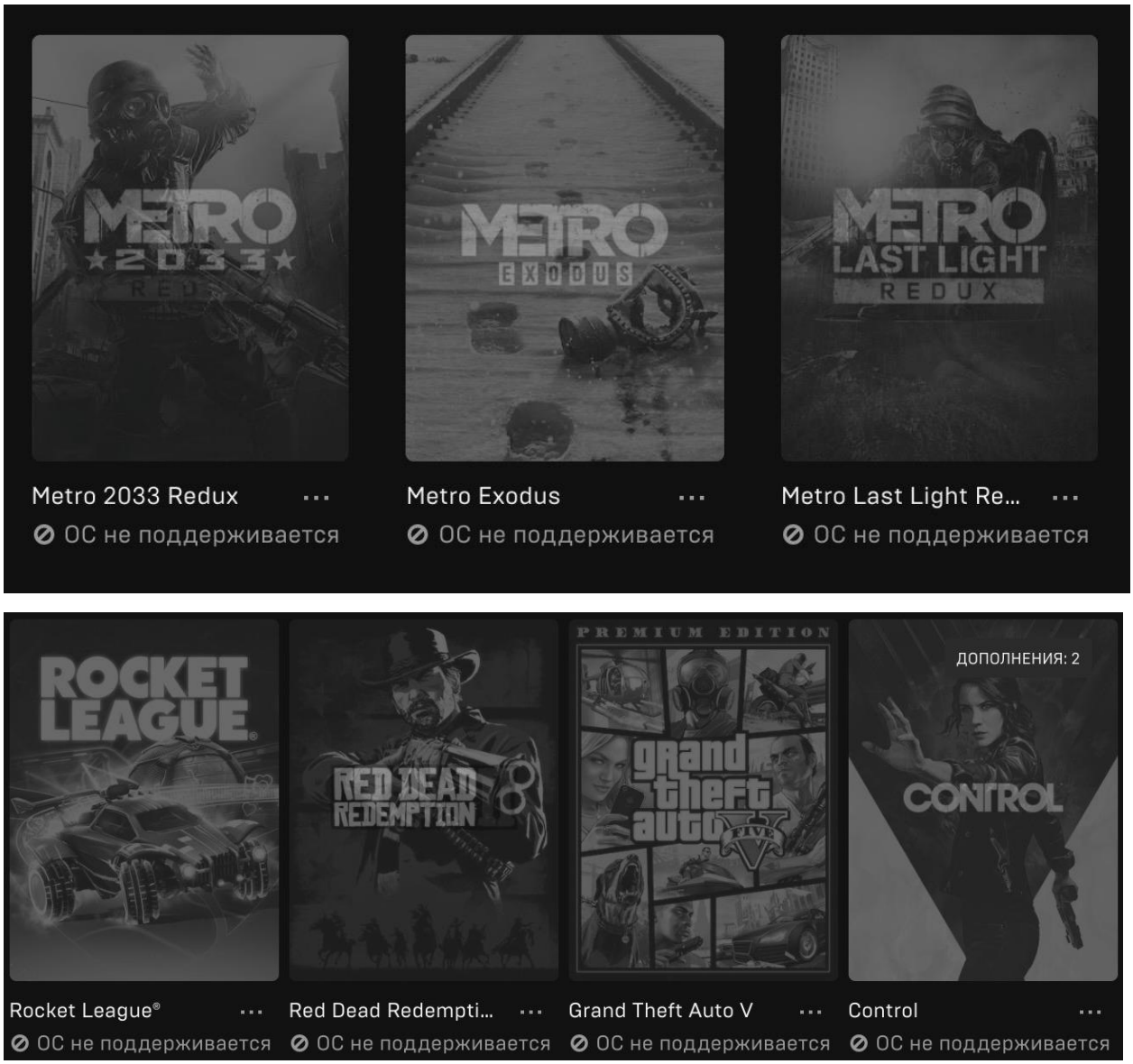
Скриншот 243. Число подсчитанных FPS в Dock без приложений на экране и без сторонних элементов в menu bar.

Если из menu bar убрать ещё и сторонние приложения, FPS без контента на экране составит 11-12. В общем, никакой особой разницы нет.

К каким выводам можно прийти? Если у Вас в приложении FPS меньше 12, вам Quartz Debug в любом случае покажет 12 FPS, если мониторинг производится через Dock, и в любом случае покажет 25+, если мониторинг производится через отдельное окно. Можно ли в таком случае верить этому счётчику FPS? Только если FPS сильно выше 25. Информация на экране Macbook не может обновляться быстрее, чем 60 раз в секунду, поэтому если FPS составляет 50-60, очевидно, что влияние Quartz Debug мало. Quartz Debug обновляет у себя на экране информацию с определённой периодичностью, примерно раз в каждые 42 мс. Очень странное совпадение должно возникнуть, чтобы игра и счётчик обновляли контент всё время с одинаковой периодичностью и FPS составил 50-60, когда на деле 25-35.

А как быть с приложениями и играми, где FPS ниже 25, т.к. их Macbook не тянет? Нам Quartz Debug нужен только для замера FPS в играх без своего счётчика. Те же, что сильно грузят GPU, имеют свой счётчик. Всё будет ок.

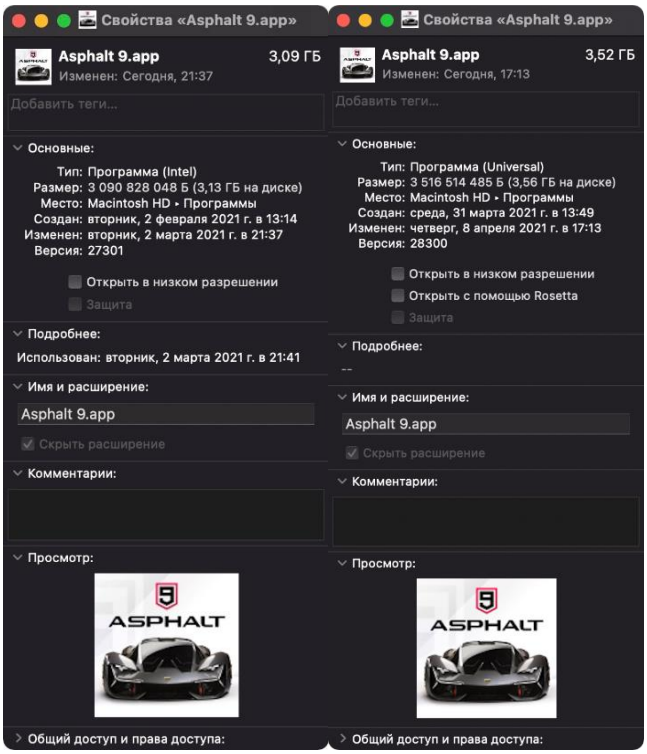
Может возникнуть у Вас вопрос по тем играм, которые мы тестируем на Macbook Pro с Apple M1. Мол, почему мы не протестировали те же игры, которые тестируем обычно на Windows-ноутбуках. Вот ответ:



Скриншот 244. Отсутствие поддержки macOS тестируемыми нами обычно играми из Epic Games Store.

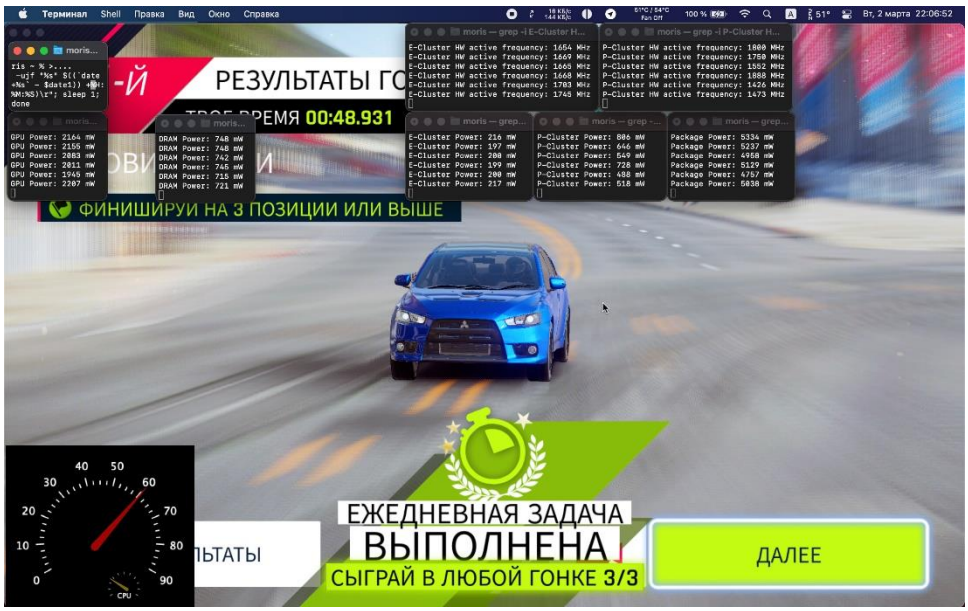
Проблема не только в том, что нам сложно тестировать игры из-за отсутствия полноценного аналога у MSI Afterburner, который бы всю необходимую информацию сразу на экран выводил. Проблема также заключается и в том, что macOS – не игровая платформа, AAA игр под неё практически нет. Если разработчики считают, что расходы на адаптацию его игры под macOS и графический API Metal не окупятся, они и не будет ничего под macOS портировать. Раньше в этом не было проблемы, ведь на Mac с Intel можно было установить Windows, но сейчас ещё и требуется адаптация игры и под ARM, не только x86. Чего уж, будем тестировать то, что можно.

10.2. Asphalt 9.



Скриншот 245. Версия Asphalt на момент тестирования в марте (слева) и на момент написания статьи (справа).

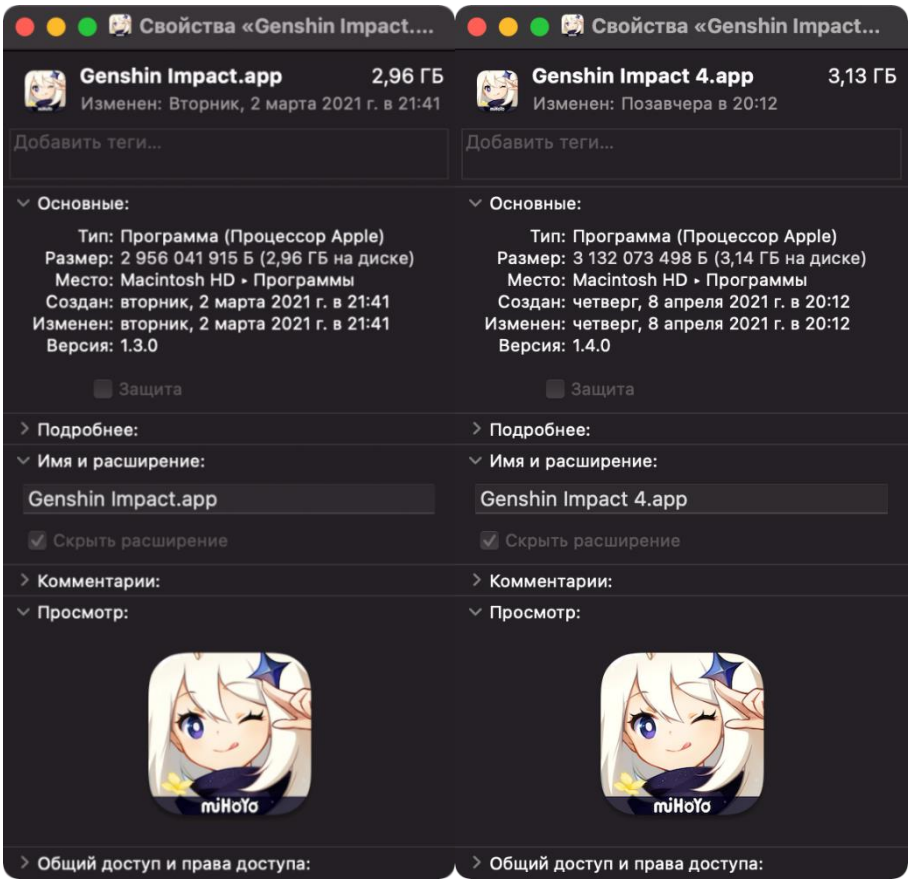
Я тестировал Asphalt 9 в марте, когда ещё не вышла его нативная версия под Apple M1. Ставил его из App Store для Mac. После релиза нативной версии на M1 хуже точно не стало, а то и стало лучше.



Видео 11. Мониторинг FPS в Asphalt 9 (Rosetta 2). Максимально доступные настройки графики

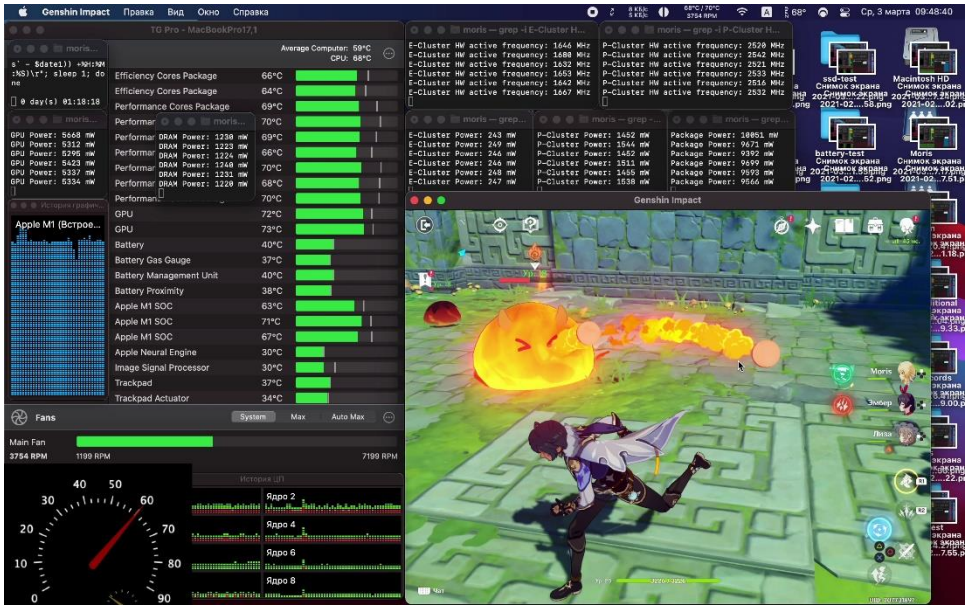
Выставил максимальные настройки графики, кроме сглаживания, т.к. оно было недоступно как до обновления (релиза натива под M1), так и после. Энергопотребление во время игры – около 5 Ватт. Разумеется, с такой задачей M1 легко справился, стабильные 60 FPS, так что идём дальше.

10.3. Genshin Impact.



Скриншот 246. Версия Genshin Impact на момент тестирования в марте (1.3.0) и после в апреле (1.4.0).

Я тестировал версию игры 1.3.0, а недавно вышло обновление 1.4.0.

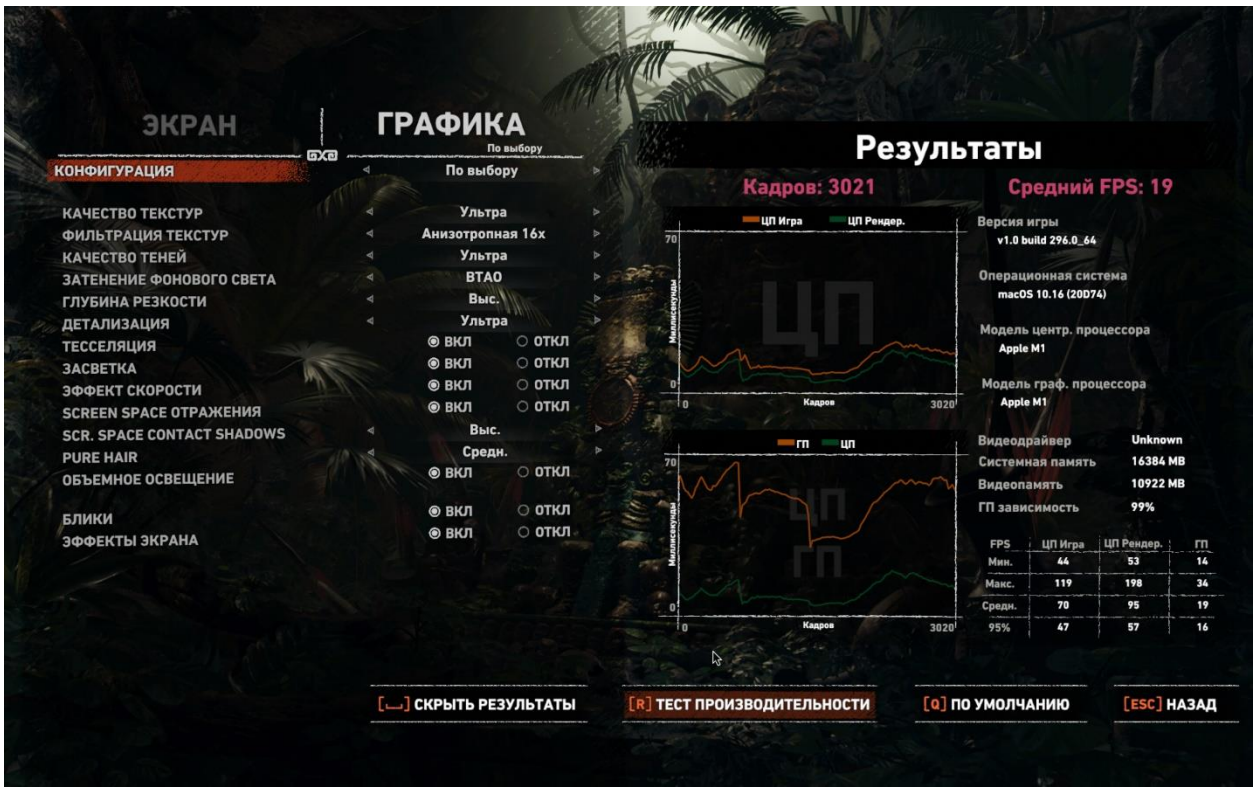


Видео 12. Мониторинг FPS в Genshin Impact (версия 1.3.0 для iPadOS). Настройки: максимальные для iPad.

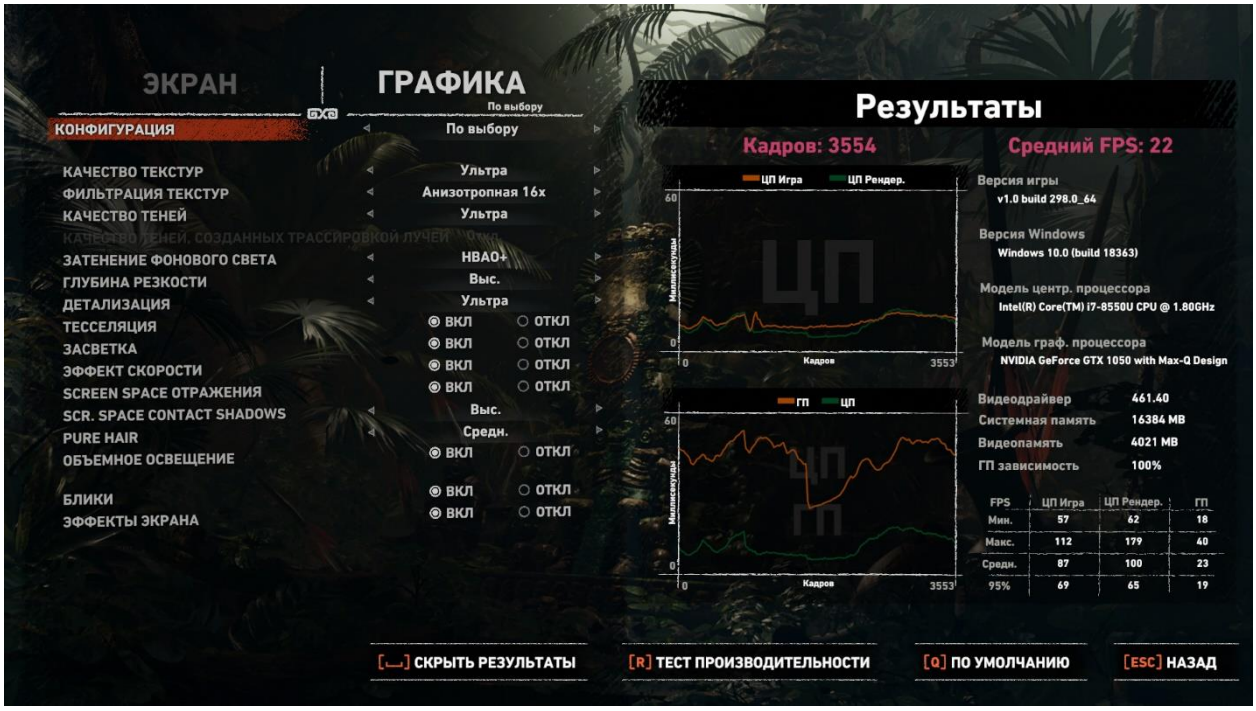
И с версией 1.3.0, и с версией 1.4.0, которую я позже проверил, видим стабильные 60 FPS, энергопотребление Apple M1 во время игры в среднем 10 Ватт, скорость вращения кулеров в пике – 3800 об/мин, в среднем – 2000-3000 об/мин. С геймпадом DualShock 4 играть было вполне комфортно.

10.4. Shadow of the Tomb Raider (2018).

Это один из немногих AAA проектов, доступных и на Windows, и на macOS, и нагружает он видеоадаптер значительно, под стать AAA проекту.



Скриншот 247. Macbook Pro (M1, 20W). Результат бенчмарка в SofTR. 1080p, макс. (вручную) настройки графики.



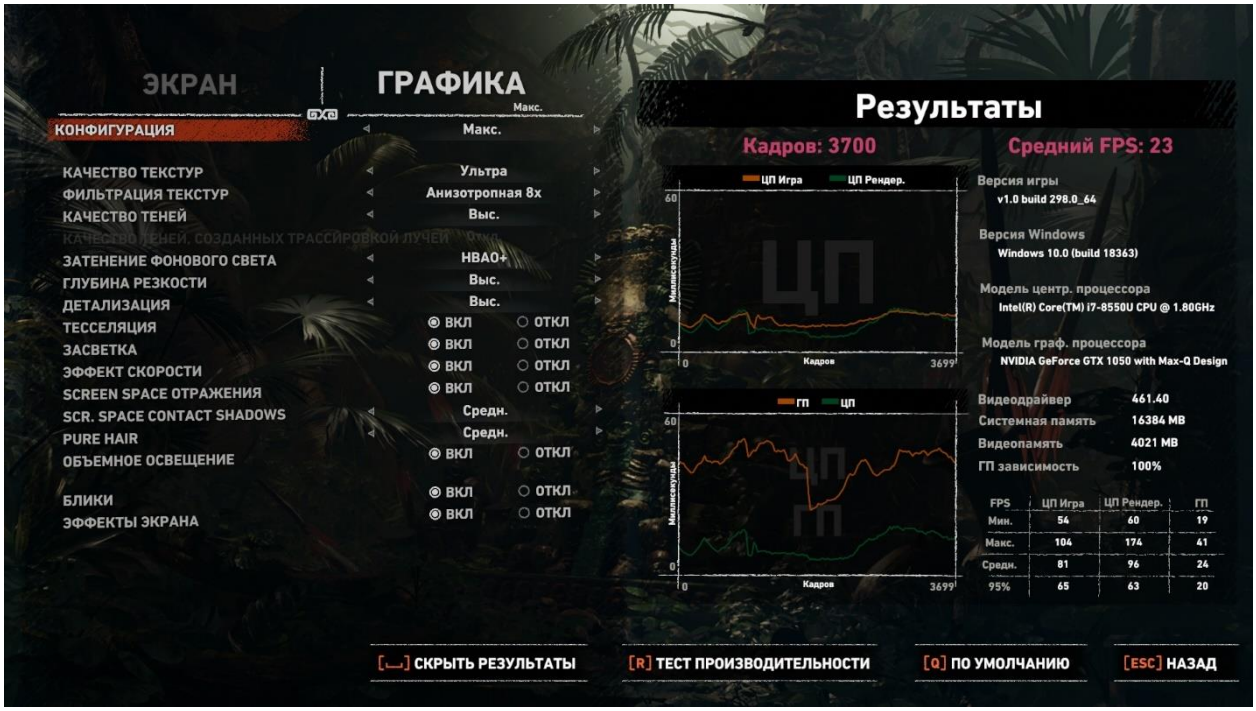
Скриншот 248. Xiaomi GTX (i7 8550U, 25W). Результат бенчмарка в SofTR. 1080p, макс. (вручную) настройки графики.

В этой игре я сравнивал результаты со своим Xiaomi Mi Notebook Pro GTX. Настройки графики: максимальные для macOS.

На Windows максимальное сглаживание – SMAA4x, но на macOS оно не доступно, поэтому я сравнивал со сглаживанием SMAAT2x оба аппарата.

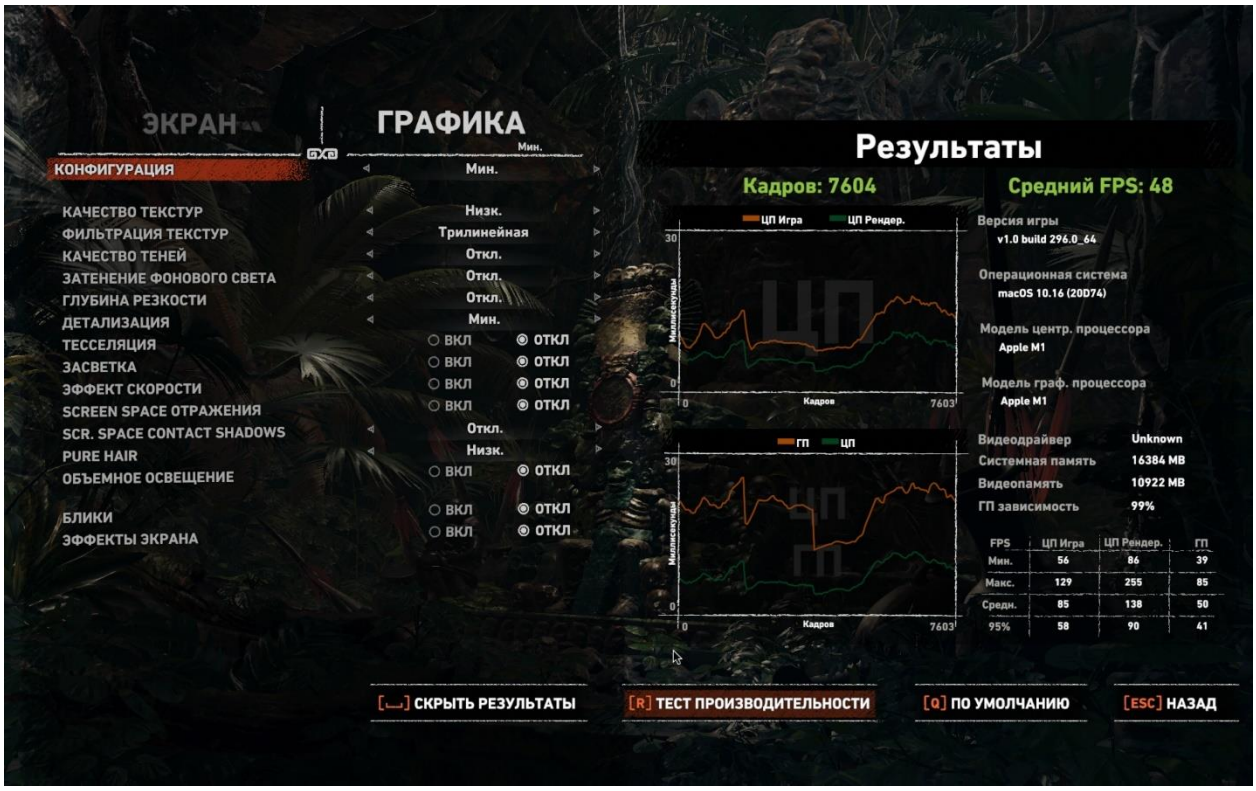


Скриншот 249. Macbook Pro (Apple M1, 20W). Результат бенчмарка в SofTR. 1080p, макс. пресет настроек графики.

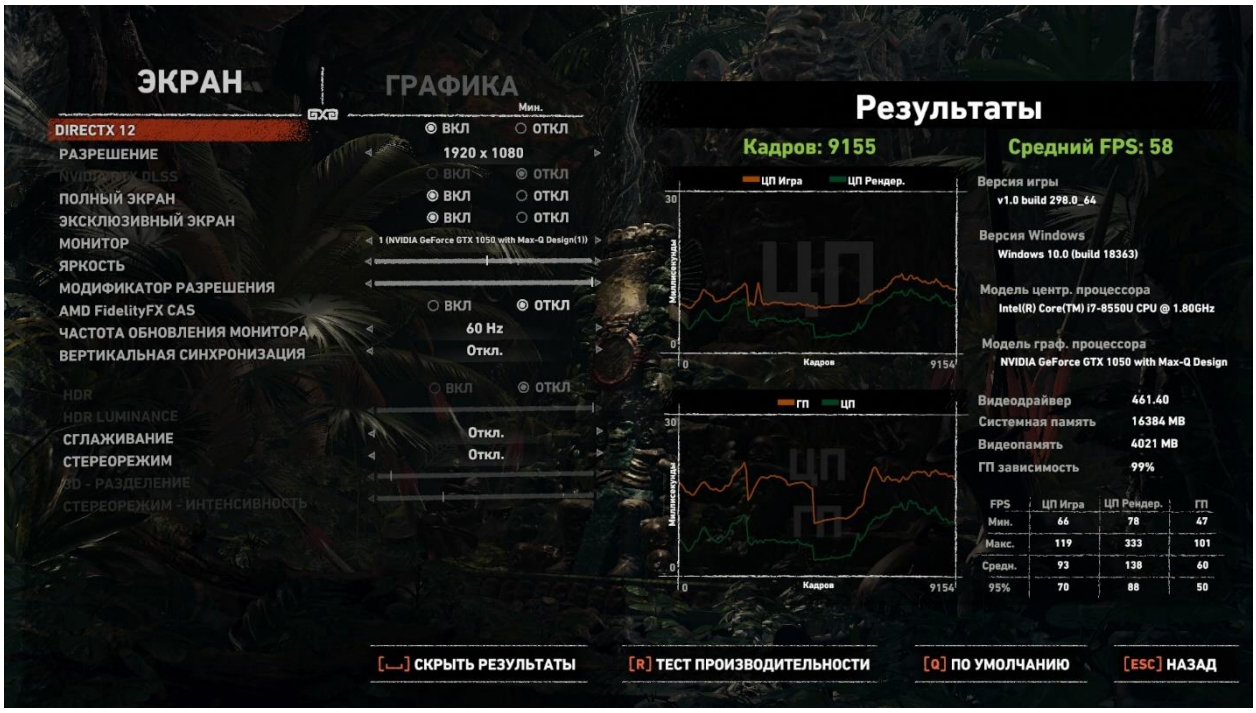


Скриншот 250. Xiaomi Pro GTX (i7 8550U, 25W). Результат бенчмарка в SofTR. 1080p, макс. пресет настроек графики.

Что с вручную установленными максимальными настройками графики, что с макс. пресетом и сглаживанием TAA (стандартное), разница всего 3 FPS (13-15%). С учётом Rosetta 2 и разницы в API Metal и DirectX, CPU+GPU в Apple M1 выходят не ниже уровнем, чем i7 8550U + GTX 1050 Max-Q.



Скриншот 251. Macbook Pro (M1, 20W). Результат бенчмарка в SofTR. 1080p мин. (вручную) настройки графики.



Скриншот 252. Xiaomi GTX (i7 8550U, 25W). Результат бенчмарка в SofTR. 1080p, мин. (вручную) настройки графики.

Если настройки графики задать вручную самые минимальные (ниже минимального пресета), разница уже существеннее, целых 10 FPS в среднем (17-21%). В качестве наименьшего 1% FPS возьмём меньшее из значений в «мин», а в качестве макс. 1% – минимальное в «макс», т.е., если, к примеру, ЦП тянет 200 FPS, а ГП – только 100, увидим мы в действительности только 100, т.к. бутылочным горлышком тут выступает ГП. Так и занесём в таблицу.

10.5. Rise of the Tomb Raider (2015).

Тут аналогичная ситуация с Shadow of The Tomb Raider.



Скриншот 253.Macbook Pro 13 (M1, 20W). Результат бенчмарка RofTR. 1080p, низкий пресет настроек графики.



Скриншот 254. Xiaomi Pro GTX (8550U, 25W). Результат бенчмарка RofTR. 1080p, низкий пресет настроек графики.

С макс. настройками по среднему кадру разница минимальна, поэтому привожу для сравнения скриншоты низким пресетом графики. Напоминаю, что обычно одни и те же компьютеры Mac с одним и тем же железом имели на Windows с API DirectX FPS выше, чем на macOS с API Metal. А тут у нас FPS должен быть по идее ещё ниже, т.к. Macbook на M1 задействует Rosetta 2 для трансляции кода игры из x86_64 в ARM64. А разница по итогу всего 7 кадров в секунду с низким пресетом графики. GPU в M1 впечатляет.

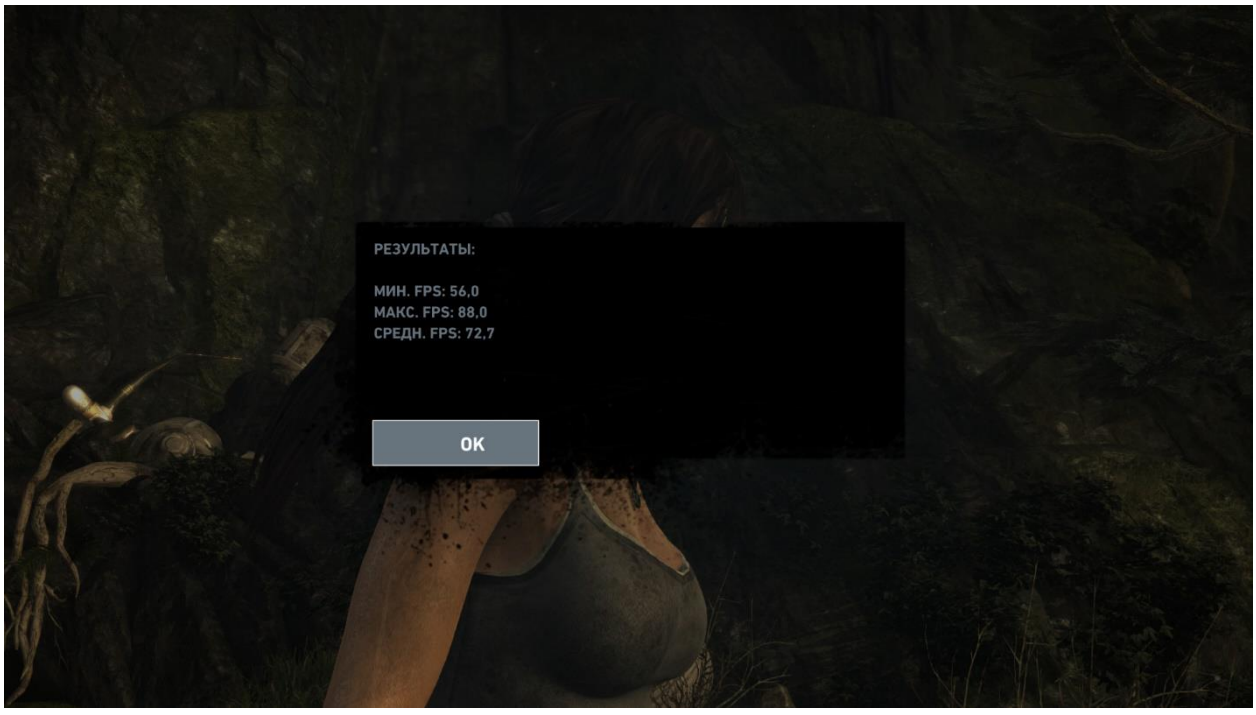
Бенчмарк в Rise of the Tomb Raider (RotTR) отражает наименьший 1% FPS и наибольший 1% FPS для каждой отдельной локации.

В таблицу мы занесём минимальное значение из всех 3 локаций для наименьшего 1% FPS, чтобы было представление о том, на что в худшем случае стоит рассчитывать во всех локациях, а в качестве наибольшего 1% FPS (или макс. FPS) занесём самое высокое значение среди 3 локаций.

10.6. Tomb Raider (2013).



Скриншот 255. Macbook Pro 13 (M1, 20W). Бенчмарк Tomb Raider. 1080p, превосходный пресет настроек графики.



Скриншот 256. Xiaomi Pro GTX (i7 8550U, 25W). Бенчмарк Tomb Raider. 1080p, превосходный пресет настроек графики.

С превосходным пресетом графики в 1080p в Tomb Raider (2013) разница уже серьёзнее: 20 кадров в секунду. В целом это вполне объяснимо различием в FPS на macOS и Windows с одним и тем же железом (не в пользу macOS) и влиянием Rosetta 2. Тут никаких сюрпризов.

10.7. Сравнительная таблица и выводы по играм.

Сведём все данные в сравнительную таблицу:

PC model			Xiaomi Mi Notebook Pro GTX			Asus ROG Strix G15			Apple Macbook Pro 13 2020 (M1)		
CPU			Intel Core i7 8550U (25W)			AMD Ryzen 9 5900HX (80W)			Apple M1 (20W)		
GPU			NVidia GTX 1050 Max-Q (30W)			NVIDIA RTX 3070 Mobile (115W)			Apple M1 (10W)		
	Res	Quality	Min 1% FPS	Avg FPS	Max (1%) FPS	Min 1% FPS	Avg FPS	Max (1%) FPS	Min 1% FPS	Avg FPS	Max (1%) FPS
Shadow of the Tomb Raider	1080	Custom Highest SMAAT2x	18	22	40	70	95	134	14	19	34
		Highest	19	23	41	70	95	130	16	20	37
		High	22	27	48				17	22	39
		Medium	25	31	50				17	22	37
		Low	30	38	62				25	33	57
		Lowest	38	47	74				31	38	67
		Custom Lowest	47	58	101				39	48	85
Rise of the Tomb Raider	1080	Custom Highest	5	16	30	33	79	157	4	15	107
		Very High	13	37	61	52	116	194	13	36	91
		High	8	43	78				11	38	99
		Medium	27	47	76				13	41	145
		Low	16	59	109				14	52	147
		Lowest	24	70	154				18	61	150
		Custom Lowest	34	75	149				17	65	152
Tomb Raider 2013	1080	Custom Highest	18	26	38	90	128	166	11	15	19
		Ultimate	39	52	64	176	223	258	22	30	38
		Ultra	56	73	88				44	51	59
		High	78	96	112				87	98	114
		Medium	110	129	146				85	100	118
		Low	140	171	194				80	86	100
		Custom Lowest	142	173	192				78	86	98

Таблица 7. Сравнительная таблица с результатами тестов ноутбуков в играх.

Выводы по играм: это, конечно, не игровой ноутбук. Да, на деле мы видим более низкий FPS из-за различия адаптации под API Metal и API DirectX, а также из-за влияния Rosetta 2, и, если сделать скидку на это, выходит, что связка CPU и GPU в Apple M1 по голой производительности не уступает i7 8550U и GTX 1050 Max-Q в моём Xiaomi Mi Notebook Pro GTX.

Вот ещё: длительные тесты своего Xiaomi я больше не буду проводить. Во время тестов в играх он от перегрева раза 3 вырубился и после он минут 10 не реагировал на кнопку питания. Я не хочу сжечь свой рабочий ноутбук.

11. Камера.

Многие жалуются на камеру в Macbook Pro. Некоторые говорят, что скоро владельцев Macbook будут узнавать по качеству видеосвязи. Так ли?



Скриншот 257. Macbook Pro 13 2020 (Apple M1). Снимок на камеру через Photo Booth.



Скриншот 258. Xiaomi Mi Notebook Pro GTX. macOS (Hackintosh). Снимок на камеру через Photo Booth.

Если сравнивать оба устройства на macOS, Xiaomi остаётся далеко позади. Возможно, у Xiaomi это косяк хакинтоша, проверим ещё Windows.



Скриншот 259. Xiaomi Mi Notebook Pro GTX. Windows. Снимок через стандартное приложение камеры. SDR.

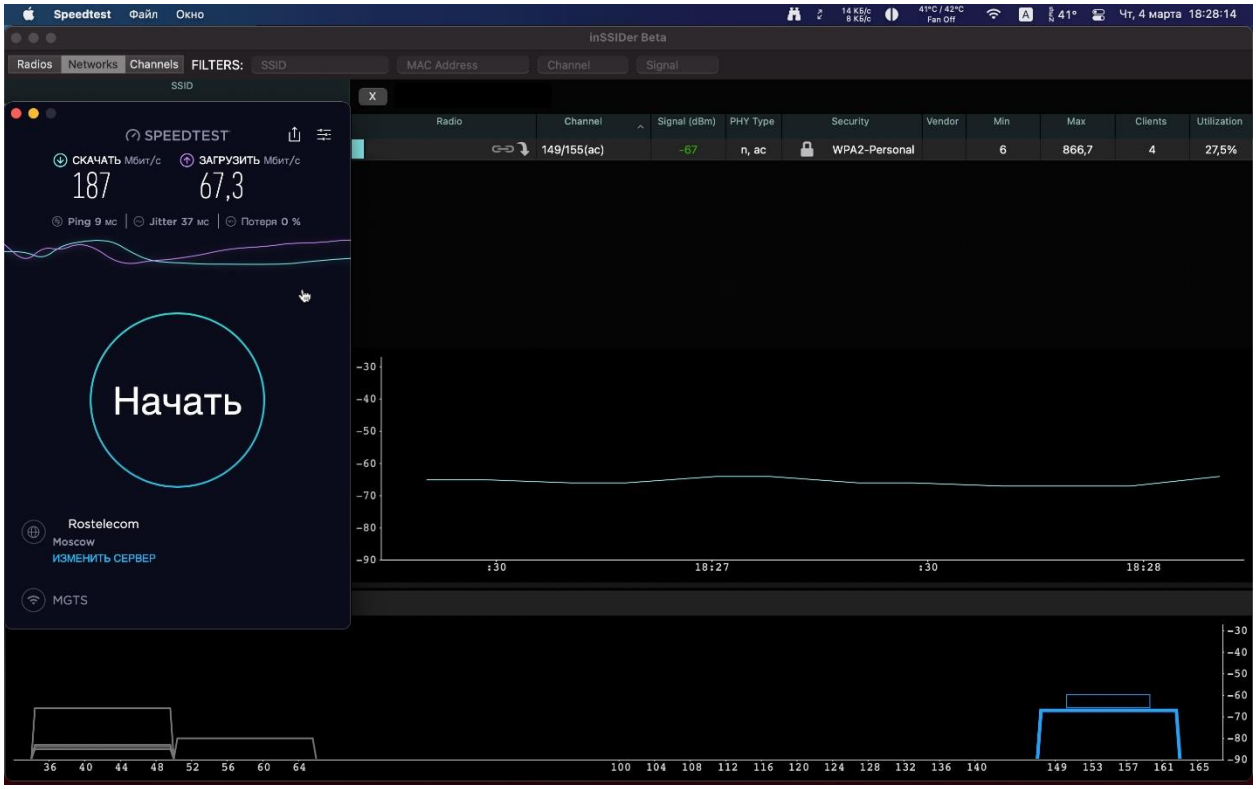


Скриншот 260. Xiaomi Mi Notebook Pro GTX. Windows. Снимок через стандартное приложение камеры. HDR.

Xiaomi и рядом не стоит с Macbook по части веб-камеры. У MBP с M1 лучшая веб-камера среди тех ноутбуков, что я обозревал на сайте. Нет, у него камера не так уж и крута, просто у остальных она хуже. А в сравнении с каким ноутбуком камера то плоха в макбуке? Да, у смартфонов фронтальные камеры лучше, но среди ноутбуков я не вижу конкурентов. В чём вопрос то?

12. Wi-Fi.

Видел в ролике у Романа Ястреба жалобу на Wi-Fi. Вот этого я понять не могу. Тестировал с Xiaomi Mi Wi-Fi Router 3G (Wi-Fi 5). Результаты ниже.

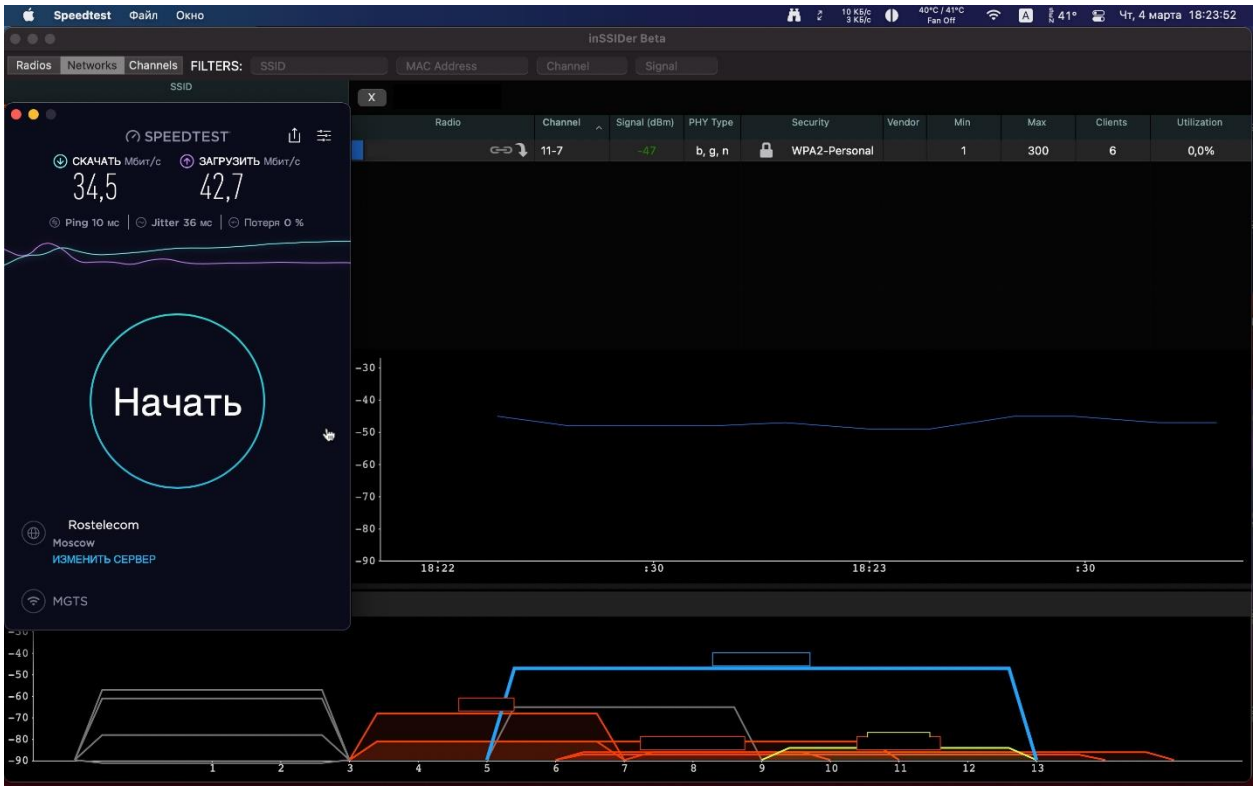


Скриншот 261. Macbook Pro 13 2020 (Apple M1). Wi-Fi 5 GHz.



Скриншот 262. Xiaomi Mi Notebook Pro GTX. Wi-Fi 5 GHz.

У моего рабочего ноутбука Xiaomi на том же самом столе, т.е. в том же удалении от Wi-Fi 5 роутера, скорость приёма почти в 2 раза ниже. Macbook намного увереннее держит сигнал. Это лучший результат в моих обзорах.



Скриншот 263. Macbook Pro 13 2020 (Apple M1). Wi-Fi 2.4 GHz.

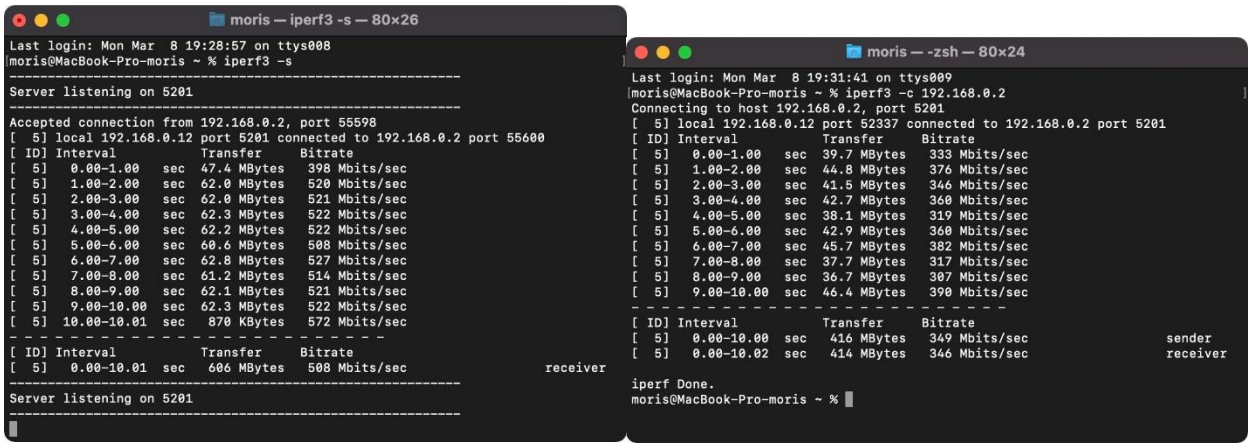


Скриншот 264. Xiaomi Mi Notebook Pro GTX. Wi-Fi 2.4 GHz.

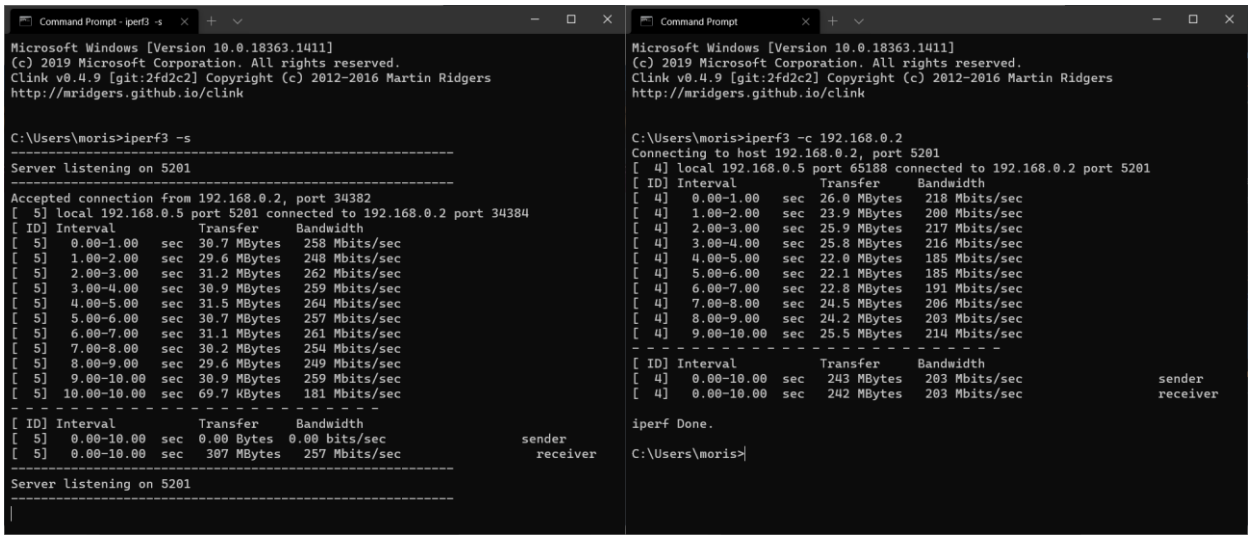
По части сигнала 2.4 ГГц они +- сопоставимы. У Xiaomi выше скорость приёма, а у Apple выше скорость отдачи, тут особой разницы не увидите. Но меня поражает то, с какой скоростью при равном удалении работает Wi-Fi в Macbook на частоте 5 ГГц. А на что тут жаловаться то? Я понимаю, что это не то, каким должен быть Wi-Fi 6 с поддержкой 160 МГц полосы, но у меня ещё ни один ноутбук с Wi-Fi 5 точкой доступа так хорошо не работал.

Читая статейку с обзором одного ноутбука на портале notebookcheck, я запримetил, что там для теста пиковой скорости Wi-Fi соединения используется утилита iperf3. Ну и я подумал: почему бы тоже не проверить?

Для тестирования скорости отдачи, я запустил сервер iperf3 на одноплатном микрокомпьютере RaspberryPi4, подключенному к моему роутеру по проводу (Ethernet, RJ-45, витая пара, зовите, как хотите) и подключился к нему с ноутбуков Apple и Xiaomi. Чтобы протестировать скорость приёма, наоборот, запустил сервер на ноутах. Всё просто, опробуем.



Скриншот 265. Macbook Pro 13 2020 (Apple M1). Скорость приёма и отдачи по iperf3.



Скриншот 266. Xiaomi Mi Notebook Pro GTX. Скорость приёма и отдачи по iperf3.

Тестировал через 1 стену на частоте 5 ГГц в 1.5 метрах от Wi-Fi 5 роутера Xiaomi. Я понимаю жалобу, если у Вас топовый Wi-Fi 6 роутер, и Вы хотите получить максимум скорости от него, но в случае с Wi-Fi 5 у Macbook скорость приёма в 2 раза выше, чем у Xiaomi, и это самая высокая скорость приёма Wi-Fi сигнала среди всех ноутбуков, что мы тестировали с моим роутером. У меня, как и у многих, Wi-Fi 5 роутер и с ним Macbook потрясен.

13. Субъективные впечатления и выводы.

~~Если меня вкратце спросить, что я думаю о Macbook Pro на M1, я отвечу, что он не стоит своих денег и лучше взять Macbook Air на M1.~~

~~Почему Air? Ужасное охлаждение в Pro. Могу ошибаться, но кроме RedmiBook 14 у нас не было подобных прецедентов: охлаждение ноутбука не способно рассеять тепловыделение в ± 20 Ватт. В чём тогда вообще смысл активного охлаждения? Зарубежные блоггеры~~ Linus Tech Tips~~,~~ Max Tech ~~и другие уже доказали, что Air с дополнительно проклеенной термопрокладкой между теплораспределительной площадкой у процессора и корпусом по производительности мало уступает Pro. Эта разница не стоит шума Pro с M1.~~

~~Оба ноутбука (Air и Pro) под серьёзными нагрузками греются до 99-101 градусов~~. Ни о какой долговечности ноутбука при использовании в тяжёлых задачах говорить не приходится. У компьютеров Apple нет режимов «холодный» или «комфорт», как у других производителей. Они всегда работают в режиме максимальной производительности. У Вас просто нет инструментов для ограничения тепловыделения SoC, так что Вам никак не спастись от перегрева SoC, его последующей деградации и высоковероятного выхода из строя компонентов. К сожалению, тут нет какого-то решения.

~~Если уж греться будет что один ноутбук, что другой, и разница по части производительности минимальна при наличии термопрокладки (только не самой дешёвой) у Air, то я не вижу смысла тратить деньги на шумный Pro.~~

Есть лишние деньги? Лучше переплатить за 16 ГБ объединённой памяти и доп. объём SSD. От объёма объединённой памяти и объёма SSD также зависит и то, насколько быстро он будет изнашиваться, т.к. при малом объёме объединённой памяти будет задействоваться чаще файл подкачки (swap) на SSD, что увеличит число записей на SSD и повысит тем самым его скорость износа, да и SSD меньшим объёмом памяти имеет меньший и ресурс записи (TBW), так что вкупе эти 2 фактора могут привести к быстрому износу из строя SSD и невозможности дальнейшей записи на него. Уж лучше доплатить за увеличенный объём объединённой памяти (16 ГБ вместо 8 ГБ) и увеличенный объём памяти SSD. Так SSD прослужит дольше.

У меня с моделью с 256 ГБ SSD и 16 ГБ объединённой за более чем 2 месяца активного использования износ менее 1% при записанных на SSD 5 ТБ данных. У Стаса с моделью на 2 ТБ SSD и 16 ГБ объединённой памяти было записано чуть меньше 7 ТБ данных и износ по итогу также менее 1%.

SSD тут холодный, у 256 ГБ модели под нагрузкой температура не превышает 42 °C, а у 2 ТБ модели – 53 °C, поэтому за сохранность данных я бы не переживал. Вот только скорость записи на SSD у 256 ГБ модели Macbook Pro с SoC Apple M1 после заполнения памяти сопоставима с недорогими SATA SSD по типу Samsung 860/870 Evo. У модели с 2 ТБ скорость тоже падает после заполнения, но меньше: с 3 ГБ/сек до 1.2 ГБ/сек.

Т.е. даже после проседания скорости записи у SSD в модели на 2 ТБ после заполнения накопителя на 49.6%, он всё ещё остаётся самым быстрым среди всех SSD, что мы тестировали в ноутбуках ранее. Уточню, что система в фоне оптимизирует свободное пространство на SSD в том числе и с помощью уплотнения уже записанной информации. Таким образом, в однобитном режиме, т.е. с максимальной скоростью, Вы ещё сможете записать некоторый объём данных (около 20%), если подождёте некоторое время после записи данных. Т.е. если разом не заполнять SSD большим объёмом данных, вероятность столкновения с этой проблемой мала. Однако, чем больше заполнен SSD, тем меньше оставшееся свободное пространство и тем меньшее пространство удастся заполнить данными в однобитном режиме в дальнейшем, т.е. тем меньше информации Вы сможете записывать на SSD с максимальной скоростью. Поэтому лучше берите модель с большей памятью.

Подробнее об этом Вы можете узнать на канале «Этот Компьютер», где автор наглядно пояснил все различия между типами памяти в SSD: SLC, MLC, TLC и QLC. В Macbook Pro с Apple M1, я предполагаю, применяется память типа TLC, т.к. скорость памяти значительно проседает относительно изначальной, но не в 10 и более раз, а в 2.5-5.5 раз, и проседает то скорость записи после заполнения накопителя примерно наполовину, а не на 10-15%.

Я понимаю, что за модель с 2 ТБ SSD мало кто будет переплачивать 60к или 80к руб., но так уж сложилось, что протестировать мы могли лишь 2

модели. Как там дела обстоят с моделями на 512 ГБ и 1 ТБ – без понятия, но вряд ли будет хуже, чем у 256 ГБ модели, так что ориентируйтесь на её скоростные показатели, решая, приобретать ли модель на 512 ГБ или 1 ТБ.

Но стоит ли вообще брать себе Macbook на M1, будь то Pro ~~или Air~~? Я считаю, что стоит, но только если у Вас достаточно денег, чтобы позволить себе купить новый Macbook, не продавая при этом свои старые аппараты.

Дело в том, что в macOS для компьютеров Mac с SoC Apple Silicon, как мы убедились в главе 2.7, некоторые системные API отличаются от таковых в macOS для компьютеров Mac с SoC Intel. Отличаются API для аккумулятора, отличаются API для блоков аппаратного кодирования и декодирования видео, поэтому в том же OBS не работает аппаратное ускорение для видео. И это не вина Rosetta 2. Сама по себе технология трансляции кода из x86_64 в ARM64 реализована хорошо у Apple. Очень сильно упрощая (надеюсь, никто не обидится на подобную аналогию), неважно, на каком языке Вы попросите человека (процессор) нарезать Вам яблоко (задействовать API) на кухне (Операционной Системе). Один человек понимает один язык, а другой понимает другой. Но проблема то не в переводе с одного языка на другой (с ним то всё в порядке), а в том, что один на кухне может пользоваться ножом, а другой вместо ножа видит ложку и не понимает, как же нарезать яблоко ложкой. Примерно так дела обстоят с аппаратным ускорением, которое, по идее, должно задействоваться одинаково на всех компьютерах Mac (с разницей в том, какой Mac какой кодек поддерживает), но у Mac на Apple Silicon всё иначе. А в случае с аккумулятором вообще дичь: Вам вместо яблока, нарезанного на мелкие кусочки, приносят яблоко, размельчённое в блендере. Это совсем другое, но да, и то, и то– яблоко, и оно размельчено.

Если бы Apple такого не сделали и реализовали полную совместимость API между Mac на Intel и Mac на Apple Silicon, тогда да, никаких вопросов бы не было. А так получается, что транслятор они написали правильно, но не обеспечили обратную совместимость API и в итоге в ряде приложений имеются проблемы, а пользователи могут всё грешить на кривую Rosetta 2, которая так-то не кривая. Проблемы тут из-за разработчиков API в Apple.

Да, Rosetta 2 не транслирует AVX-инструкции, т.к. регистров на 256 бит, необходимых для их корректной работы, процессоры с архитектурой ARM попросту не имеют. Но то, что она транслирует, она транслирует верно. Да и практически всё ПО, задействующее AVX-инструкции, умеет в целом работать и без них, просто будут операции выполняться медленнее на x86_64 системах. Все вопросы только к тем программистам Apple, которые для Mac с Apple Silicon решили изменить параметры внешнего доступа к API, а не реализовать просто на более низком уровне системы обработку одних и тех же вызовов по API по-разному на Mac с Apple Silicon и Mac с Intel. Так ещё и с приложениями для iOS и iPadOS не обеспечена совместимость полностью: одни приложения работают, а другие – нет. В итоге у нас ни рыба, ни мясо.

Технологии виртуализации у Apple Silicon свои, они отличаются от таковых у Intel, поэтому тот же Docker сейчас переписывают под Apple Silicon. Он уже представлен в версии RC3, т.е. скоро можно ожидать и релиза стабильной версии, но пока косяки вполне возможны. Впрочем, то, что я проверял, пашет нормально уже сейчас.

Есть проблемы с ПО: в главе 2.7 программа Benchmark от Kavars на одном экземпляре Mac с M1 запустилась, а на другом – нет.

Решился вопрос в главе 2.7 перекомпиляцией программы из исходного кода под «My Mac» вместо «Any Mac». В чём причина проблемы? А вот хз.

Если Вы занимаетесь программированием и пишете ПО на Swift и Python, в принципе Вы вполне можете взять себе новый Mac на Apple M1, он для этих задач вполне подойдёт. А вот тем, кто пишет ПО на Obj-C, имея в проекте код, задействующий уникальные архитектурные особенности x86_64 платформы, или же имеющий в своём составе библиотеки, полагающиеся на такие особенности, я бы брать Mac на Apple M1 рекомендовал только в том случае, если у них имеется резервная рабочая станция для работы с проектами, т.к. вполне вероятно, что с проблемами с M1 Вы столкнётесь. Для тех, кто пишет код на Rust, C, C++ и Java рекомендация та же.

По части работы с видео и графикой: тут спорно. eGPU к Mac на M1 вы не подключите из-за отсутствия драйверов. Я бы не переживал особо из-за

этого, т.к. интегрированный видеоадаптер в Apple M1 уже сопоставим с GTX 1050 Max-Q от NVidia по своей производительности, а в следующих SoC, скорее всего, GPU будет ещё производительнее. И вполне вероятно, что Apple релизнут в скором времени и свои собственные eGPU с драйверами под ARM. Если Вам хватает 1050 Max-Q, можете уже сейчас взять Mac с M1.

Касательно игр: если Ваши игры задействуют API Metal и работают через Rosetta 2 или нативно, то почему бы и нет. Вы можете от M1 ожидать ± 60 FPS с низкими настройками графики едва ли не во всех доступных играх.

Если Вам нужны приложения, доступные только в Windows: забудьте. На данный момент есть всего 2 способа запуска Windows приложений: CrossOver и Parallels Desktop. И оба этих способа на данный момент на Mac с Apple Silicon работают ужасно. С первым то понятно, он сам под Apple M1 не адаптирован ещё, а вот второй мы проверяли в Technical Preview (сейчас доступна стабильная сборка версии 16.5) и через него можно запустить только Insider Preview версию Windows 10 on ARM, и в таком сценарии повезёт, если будет работать то, что Вам нужно. Insider Preview версия Windows 10 сама по себе весьма нестабильна, на мой взгляд, поэтому даже релиз стабильной версии Parallels Desktop не особо поправит дело. А стабильную версию Windows 10 on ARM Вы себе не установите, т.к. Microsoft её распространяет лишь для OEM-производителей ноутбуков и планшетов. Вы не можете просто пойти и купить себе «коробочное» издание Windows 10 on ARM, Вы не можете загрузить установщик операционной системы и с сайта Microsoft. Т.е. если Вы и запустите Windows через Parallels Desktop, это будет только нестабильная Insider Preview версия, с который Вы никак не сможете быть уверены в том, как поведёт себя Ваш ноутбук в тот или иной момент. Уж лучше просто иметь в запасе ПК с Windows или старый компьютер Mac с SoC Intel, и использовать его для запуска Windows. На Mac с SoC Intel через BootCamp легко ставится нативная Windows, проблем не будет. Смысла в «угадайке» (будет или не будет работать) я не вижу на M1.

Если Вы стримите видео на YouTube, Twitch, WASD, или любую другую платформу, дождитесь выхода стабильной версии OBS 27 или

последующей, где будут устранены проблемы с аппаратным ускорением кодирования видео в H.264, которые разработчикам создала Apple своими API. Вы можете взять Mac и сейчас, но пока аппаратное ускорение работать не будет.

Можно ли его брать для работы в Blender и 3D-моделирования в целом? Если Ваше ПО работает на Apple Silicon нативно, или, хотя бы, корректно пашет при трансляции через Rosetta 2, да, берите, будет всяко лучше аналогичного по цене Macbook прошлого поколения на Intel, и будет даже производительнее имеющихся в продаже Macbook Pro с SoC Intel и 4 портами Thunderbolt 3, которые стоят значительно дороже. Проверяйте, работает ли Ваше ПО на Apple M1 нативно на IsAppleSiliconReady, а если нет – есть и проблемы с трансляцией через Rosetta 2. У того же Blender, пока я писал эту статью, вышла бета-версия 2.93, сборка c75b2019e101 от 19 апреля, и она нативно запускается на Apple Silicon. Но к этому моменту я уже отформатировал и сдал Macbook на M1, поэтому я не могу проверить, как работает эта версия. Да и смысла нет сейчас проверять бета-версию и по ней делать какие-то выводы, особенно касательно стабильности работы.

Меня поначалу бесил TouchBar в Macbook Pro, но я нашёл на просторах интернета утилиту Bar None, которая блокирует реакцию сенсорной панели у TouchBar на Ваши прикосновения, пока Вы не нажмёте на клавишу «FN». Не могу всем подряд советовать эту утилиту, т.к. не изучал её исходный код, да и Вы его не изучите, раз уж он закрытый, поэтому нельзя гарантировать того, что утилита не ворует какие-либо Ваши личные данные. Но она работает. И мне она сильно помогла. Понимаете ли, проблема TouchBar в том, что он реагирует на мельчайшие касания. Вот набираете Вы текст на клавиатуре, вводите цифры, ненароком пальцы задевают кнопки на TouchBar, и Вы уже в меню приложений, или Вы уже переключили трэк, или Вы ещё какое-либо действие выполнили, которого не желали. В случае с обычными кнопками мало их коснуться, нужно их прожать.

~~Собственно, из-за отсутствия TouchBar лично я бы также отдал предпочтение Air, но у меня не возникло проблем с использованием~~

~~клавиатуры на Macbook Pro с M1 после того, как воспользовался утилитой~~ Bar None~~. Вполне себе вариант, но, опять же, я не вижу смысла брать Macbook Pro с Apple M1, который от Air то по большей части отличается лишь активным охлаждением и TouchBar, чтобы затем решать проблемы с этим самым TouchBar. Смысл? Возьмите тогда Air, а вопрос охлаждения решите термопрокладкой, ещё и решение будет совсем бесшумным.~~

UPDATE: *Большое спасибо Роману Ястребу за его новый видеоролик, в котором он, разбирая видео Linus`a, ткнул в ошибку, которую и мы далее растиражировали в обзоре Macbook Pro на M1.*



Видео 13. Роман Ястреб: "MacBook Air M1 умрет с термопрокладкой! "

Роман Ястреб показал, почему нельзя на Air клеить термопрокладку: тепло с SoC (процессора, интегрированной графики, чипа безопасности, нейронного модуля и т.д.) таким образом будет отведено не только на корпус, а далее с корпуса вовне и рассеяно, но также тепло с корпуса будет таким образом отведено и на внутренние компоненты ноутбука, в т.ч. и на аккумулятор. Батарея ноутбука у Вас, таким образом, сильно нагреется и она очень быстро от этого деградирует, т.е. за короткий срок её износ будет и в самом деле весьма велик. В худшем случае она и вовсе вздуется.

В видео Роман проводил тест с Cinebench, который, в отличие от Blender, не грузит процессор в M1 на 100%. SoC Apple M1 в тесте Cinebench R23 потребляет всего 16 Ватт вместо 20, и выделяет оттого меньше тепла, и даже так температура аккумулятора у Романа достигала 45+ градусов, что для него крайне неблагоприятно. И это всего на 7 минут...

У меня к видео Романа только 2 поправки.

Первая поправка: даже если SSD нагреется до 70 градусов, это не критично. По-хорошему его температура не должна превышать 80-85 градусов, чтобы тот не деградировал, а вот критическая температура для SSD, как и для CPU, около 100 градусов.

Вторая поправка: 46 градусов – не критическая температура для акккумулятора. iStat Menus просто в качестве верхней планки/границы графика отображает максимальное значение, которое он зафиксировал, запрашивая у системы информацию о сенсорах каждые 4 секунды, а вот на самом графике информация обновляется лишь раз в 20-30 секунд. Так и получается, что на графике максимум – 45, а верхняя планка – 46. Ну и учтите, что значение для верхней планки округляется в большую сторону, если, например, была зафиксирована температура в 45.5 градусов. Не будет же верхней планкой в графике выставлено значение 45.5 градусов, а будет указано значение в 46 градусов, т.к. верхняя граница тут представлена всегда целым числом. Может ли аккумулятор работать при температуре 45-46 градусов? Да, может, но с течением времени при такой температуре он в лучшем случае быстро утратит ёмкость, в худшем – вздуется. Ну его...

В любом случае моя рекомендация взять Air вместо Pro и проклеить термопрокладку ошибочна. Я основывал свою рекомендацию на видео LinusTechTips и Max Tech, первого из которых и критикует Роман в видео.

Но я всё ещё рекомендую брать Air вместо Pro, т.к. для полностью бесшумного ноутбука он безумно производительный. Если сравнить его результаты на Air со скриншота на 12:24, тот всего на 15%-34% (разница возрастает в течение теста первых 4 сцен) дольше рендерит сцены в Blender, но при этом никак не шумит, тогда как Pro в этих же тестах у меня свистел весьма ощутимо. Если прирост по производительности процессора в 30% при длительных нагрузках для Вас важнее отсутствия шума, берите Pro. Про проклейку термопрокладок на Air и думать забудьте.

Информацию мы со Стасом приняли, в отчёт об ошибках её включим.

У меня было 2 варианта, как поступить с этой информацией после того, как я её принял: или вырезать из своего обзора всякое упоминание термопрокладок, или просто дополнить обзор дополнительной информацией. Если бы я выбрал первое, Вы бы никак из этого обзора не узнали, что этого ни в коем ради делать не стоит, поэтому я выбрал второе. Надеюсь, мысль я донёс, и Вы не проклеите термопрокладку на Air.

UPDATE 2: *спустя полгода после написания обзора на Macbook Pro на базе Apple M1, в сентябре 2021-го мы вместе с моим братом собрали деньжат и подарили моей сестре Macbook Air на Apple M1 (брали в версии 16/512). И спустя ещё несколько месяцев, в январе 2022-го, т.е. спустя почти год после публикации статьи, я одолжил у сестры её Macbook на время, и прогнал ещё несколько тестов. И тут вскрылся нюанс: троттлинг у Macbook Air на самом деле просто огромный, и не важно, грузим ли мы CPU, или же грузим GPU.*

Не знаю, по какой причине, но Air и Pro в троттлинг-тесте ведут себя по-разному. Один (Pro) постоянно бьётся в температурный порог в 100 градусов, а другой, как оказалось, спустя время опускает этот порог. Спустя пол часа у Air температура не превышала уже 85 градусов. С одной стороны, это круто, ведь так выходит, что он дольше прослужит, т.к. реже будет перегреваться. А с другой: а как температура снизилась то? Кулеры, что ли, возникли из неоткуда у Air? Нет, он ещё сильнее скинул частоту, чтобы сбавить тепловыделение процессора. Частота упала до 1.3 ГГц, и так ему удалось сбавить тепловыделение до 5.5 Ватт. Спустя время она обратно уже не поднималась, т.е. спустя час картина всё та же.

Понимаете разницу? У Pro процессор постоянно вжаривал на полную, на максимум который позволяла система охлаждения, сколь бы фиговой она ни была. А Air сперва урезает тепловыделение почти в 2.5 раза в сравнении с Pro, а затем и вовсе решает, что надо бы быть холоднее, и срезает его почти в 4 раза за счёт снижения частоты более чем в 2 раза. В 2, Карл!

И с троттлинг-тестом GPU ситуация оказалась точно такой же.

Я запустил на Air ту же панель настроек графики в Tomb Raider, что запускал и на Macbook Pro с M1. В первые 2 минуты FPS был около 85. Там циклически воспроизводится одна и та же сцена на заднем плане, и она неплохо так нагружает GPU. Сперва FPS был около 85, т.е. показатели в начале были те же, что и у Pro с M1.

И не имеет значения, 7 ядер у GPU или 8. В макбуке сестры 7 ядер у GPU, но, пока троттлинга нет, производительность почти как у Pro с 8.

Но, спустя полчаса, FPS упал с 85 до 40. Более чем в 2 раза, Карл! Температура также снизилась со 100 до 75 градусов, т.к. тепловыделение процессора упало до 4.5 Вамт. Т.е. Air в сравнении с Pro с M1 в задачах, где требуется высокая производительность графики, также оказывается примерно в 2 раза медленнее.

А я ещё думал, почему у сестры на Macbook Air спустя пол часа игры в Genshin Impact притормаживать начинает, словно FPS раза в 2 падает...

Вывод тут какой?Air не предназначен для длительных нагрузок. Если его грузить 10 минут и более, он будет сильно медленнее, чем Pro. Спустя пол часа разница доходит до 2 раз. И так что по части нагрузки на CPU, что по части нагрузки на GPU. Air, может быть, мог бы быть и шустрее, но Apple на Air с M1, в отличие от Pro, спустя время старается держать более низкую температуру. Поэтому, если хотите взять мак для серьёзной работы, доплатите уже 20 тыс. и возьмите себе Pro. Нам с моим братом этот Macbook Air (16/512) обошёлся примерно в 120 тыс. руб., и грустно то, что он теряет половину своей производительности спустя пол часа... Лучше бы доплатили тогда и купили бы Pro. Но тут уж извините, куда деваться.

В некоторых задачах, например, при рендеринге в Blender, разница, может быть, будет не меньше, около 30-40%, но в ряде других задач у вас производительность на Air может просесть аж в 2 раза, и это видно даже в панели настроек в Tomb Raider (2013). Ну нахер Air для серьёзной работы.

На случай, если вдруг выяснится какая-то ещё интересная информация, я опубликую её в моём Телеграм-канал по технологиям*, также как сделал и* в данном случае*. Оставайтесь на связи и не пропустите инфу.*

Можно ли всем без исключения рекомендовать Mac на Apple M1? Разумеется, нет. Вам нужно исходить из Ваших конкретных задач и оценивать лично для себя, подходит он Вам или нет. ~~Мне бы Air подошёл и, думаю, я возьму его себе, когда у меня будут на то свободные ± 120к руб. для версии с 16 GB RAM и 512 GB SSD~~ (беру назад эти слова после **UPDATE 2**).

Macbook в целом мне понравился: у него, в отличие от многих ноутбуков, которые я тестировал, под нагрузкой не глючит тачпад, он никак не пищит (только кулеры свистят тут, т.е. с системой питания всё хорошо), у него отличный звук из динамиков (только Macbook Pro 16” меня впечатлил больше своими динамиками), у него отличный экран и у него отличные материалы корпуса (в отличие от моего Xiaomi Mi Notebook Pro, у Macbook Pro на M1 корпус не стёрся об мой металлический ремешок под Mi Band 2). Да, недостатков у него хватает, как, в общем-то, у любого ноутбука из тех, что я тестировал. При покупке ноутбука всегда задавайтесь вопросом «с каким недостатками я готов мириться?». ~~Лично я готов принять косяки Macbook Air на Apple M1 (но не Pro, ну нафиг), так что я (были бы деньги) смог бы насладиться всеми его преимуществами, перейдя на него с Xiaomi.~~ *Ну его нахрен, друзья, лучше возьму себе Pro 14” или Pro 16” с M1 Pro. Я считал Air хорошим выбором. Хотя, раз он мою сестру радует, с ролью подарка он справился. Но пусть эта информация хоть кому-то принесёт пользу и вы не совершите ошибки. Если планируете сильно грузить ноутбук, берите себе Pro. Никакого Air, забудьте про него в таком случае. Я полностью меняю своё мнение о Macbook Pro. Каким бы плохим ни было его охлаждение, оно у него, хотя бы, есть, и свою задачу хоть как-то решает.*

Спасибо, что прочитали эту статью, которая по своему объёму уже тянет на книгу. Надеюсь, что Вам было интересно также, как и мне. Жду Ваших комментариев на сайте. Если нашли какие-либо неточности или нашли ошибки, обязательно дайте мне о них знать, и я их поправлю.

На этой ноте я с Вами прощаюсь до следующих обзоров.

Всего Вам доброго!