CSUN | CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP 222

DR JEFF SOFTWARE
INDIE APP DEVELOPER

COMP222

Spring 2023 | Rev 1-21-23

Computer Organization

Benchmarks & Sims

Dr Jeff Drobman

website → drjeffsoftware.com/classroom.html

email → jeffrey.drobman@csun.edu

Section

Simulators

- MIPS MARS
- ARMsim
- x86

Simulators

	CPUlator	MARS 4.5	QtSPIM 9.1.20	ARMSim# 1.91	ARMSim# 2.1
No installation required	✓	✗	✗	✗	✗
Platform	Web browser	Java JRE	Windows, OSX, Linux	.NET 3.0	.NET 3.0
Free	✓	✓	✓	✓	✓
Open-source	✗	✓	✓	✗	✗
Editor	✓	✓	✗	✗	✗
Code completion	✓	✓	n/a	n/a	n/a
Assembler	**GNU**	custom	custom	custom	**GNU**
C or other languages	✓	✗	✗	✓	✓
Debugger	✓	✓	✓	✓	✓
Breakpoints	✓	✓	✓	✓	✓
Single-step	✓	✓	✓	✓	✓
Reverse step	✗	✓	✗	✗	✗
Step over function	✓	✗	✗	✓	✓
Step out of function	✓	✗	✗	✗	✗
Modify registers	✓	✓ (except pc)	✓	✗	✗
Modify memory	✓	✓	✓	✗	✗
Show call stack	✓	✗	✗	✗	✗
Runtime calling convention checks	✓	✗	✗	✗	✗
Data watchpoints	✓	✗	✗	✗	✗
Instruction sets	MIPS32 r5 MIPS32 r6 ARMv7 Nios II	MIPS32	MIPS32	ARMv5	ARMv5
Self-modifying code	✓	✓	✗	maybe	maybe

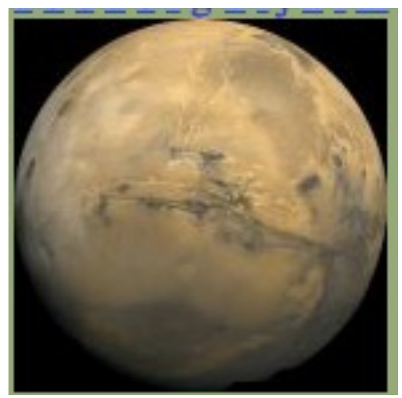
Simulators

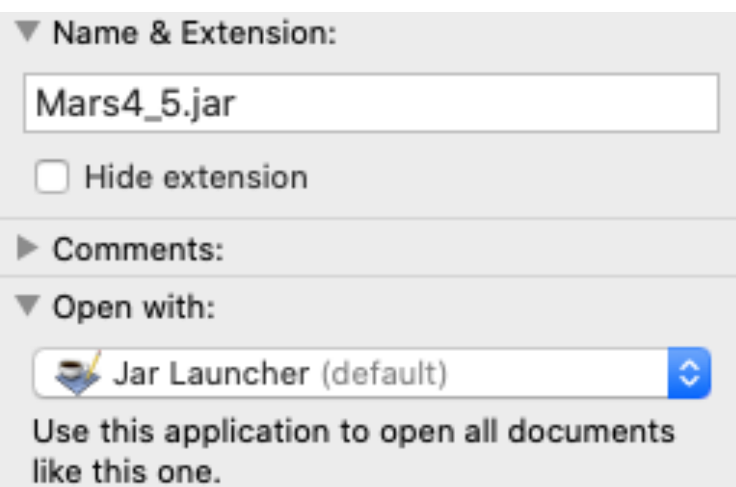
Data watchpoints	✓	✗	✗	✗	✗
Instruction sets	MIPS32 r5 MIPS32 r6 ARMv7 Nios II	MIPS32	MIPS32	ARMv5	ARMv5
Self-modifying code	✓	✓	✗	maybe	maybe
MMU	✗	✗	✗	✗	✗
FPU	✓	✓	✓	✓	✓
Memory model	4 GB flat	5 segments	5 segments	1 segment	1 segment
Maximum usable memory	2042 MB	4+4+4 MB data 4+4 MB code	4+1+0.5 MB data 256+64 KB code	64 KB data 512 MB code	96 KB data 512 MB code
I/O devices	✓	✓	✓	✓	✓
Terminal	✓	✓	✓	✓	✓
File I/O	✗	✓	✓	✓	✓
Other devices	✓	✓	✗	✓	✗
Simulation speed (Minst/second)	13	3	10	2	3

Lab

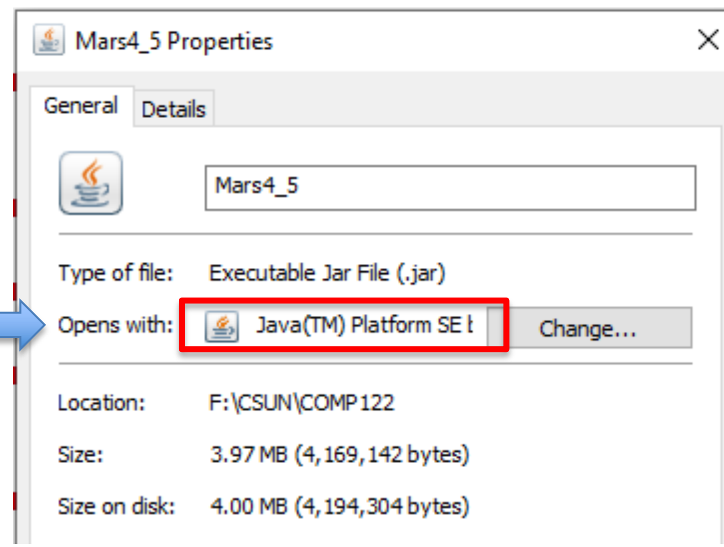
MIPS

MARS





Windows 10



CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP222

DR JEFF SOFTWARE INDIE APP DEVELOPER

MARS

MARS (MIPS Assembler and Runtime Simulator)

Missouri State UNIVERSITY

courses.missouristate.edu

Mac Desktop

https://courses.missouristate.edu/KenVollmar/MARS/download.htm

Download MARS 4.5 software! **(Aug. 2014)**

Note: Is your MARS text unreadably small? *Download and use a new release Java 9, which contains a fix to automatically scale and size AWT and Swing components for High Dots Per Inch (HiDPI) displays on Windows and Linux. Technical details.*

MARS features overview: *(List of features by version)*

- *GUI with point-and-click control and integrated editor*
- *Easily editable register and memory values, similar to a spreadsheet*
- *Display values in hexadecimal or decimal*
- *Command line mode for instructors to test and evaluate many programs easily*
- *Floating point registers, coprocessor1 and coprocessor2. Standard tool: bit-level view and edit of 32-bit floating point registers (screenshot).*
- *Variable-speed single-step execution*
- *"Tool" utility for MIPS control of simulated devices. Standard tool: Cache performance analysis tool (screenshot).*
- *Single-step backwards*

MARS

MARS (MIPS Assembler and Runtime Simulator)

MARS - Mips Assembly and Runtime Simulator

Release 4.5

August 2014

Introduction

MARS, the **M**ips **A**ssembly and **R**untime **S**imulator, will assemble and simulate the execution of MIPS assembly language programs. It can be used either from a command line or through its integrated development environment (IDE). MARS is written in Java and requires at least Release 1.5 of the J2SE Java Runtime Environment (JRE) to work. It is distributed as an executable JAR file. The MARS home page is **`http://www.cs.missouristate.edu/MARS/`**. This document is available for printing there.

As of Release 4.0, MARS assembles and simulates 155 basic instructions of the MIPS-32 instruction set, approximately 370 pseudo-instructions or instruction variations, the 17 syscall functions mainly for console and file I/O defined by SPIM, and an additional 22 syscalls for other uses such as MIDI output, random number generation and more. These are listed in separate help tabs. It supports seven different memory addressing modes for load and store instructions: `label`, `immed`, `label+immed`, `($reg)`, `label($reg)`, `immed($reg)`, and `label+immed($reg)`, where `immed` is an integer up to 32 bits. A setting is available to disallow use of pseudo-instructions and extended instruction formats and memory addressing modes.

Our guiding reference in implementing the instruction set has been *Computer Organization and Design, Fourth Edition* by Patterson and Hennessy, Elsevier - Morgan Kaufmann, 2009. It summarizes the MIPS-32 instruction set and pseudo-instructions in Figures 3.24 and 3.25 on pages 279-281, with details provided in the text and in Appendix B. MARS Releases 3.2 and above implement all the instructions in Appendix B and those figures except the delay branches from the left column of Figure 3.25. It also implements all the system services (syscalls) and assembler directives documented in Appendix B.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Mac MARS/JAR File Access

DR JEFF SOFTWARE
INDIE APP DEVELOPER

To give Full Disk Access to JAR files on macOS:

1. Go to System Preferences.
2. Click on Security and Privacy.
3. Search for 'Full Disk Access'.
4. Click on the lock at the bottom left to be able to make changes.
5. Click on the '+' icon at the bottom left of the FDA panel and a Finder prompt will appear.
6. Go to System/Library/CoreServices/JavaLauncher.app
7. Select the JavaLauncher.app and click 'Open'

It works!

MARS

MARS (MIPS Assembler and Runtime Simulator)

Registers

Mars — MARS 4.5

File Edit Run Settings Tools Help

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000

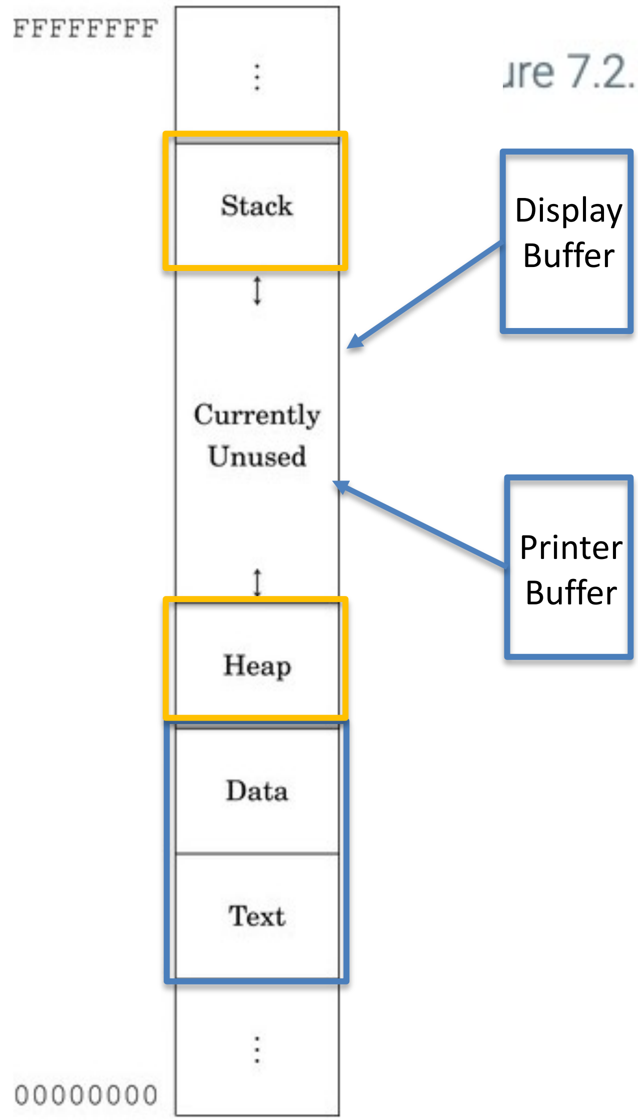
Registers | Copro

Name	Float
$f0	0x00000000
$f1	0x00000000
$f2	0x00000000
$f3	0x00000000
$f4	0x00000000
$f5	0x00000000
$f6	0x00000000
$f7	0x00000000
$f8	0x00000000
$f9	0x00000000
$f10	0x00000000

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Memory Segments



Typical memory layout for a program with a 32-bit address space.

ure 7.2.1: Object file (COD Figure A.2.1).

A UNIX assembler produces an object file with six distinct sections.

Object file header	Text segment	Data segment	Relocation information	Symbol table	Debugging information

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

MARS

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

MARS (MIPS Assembler and Runtime Simulator)

Memory Map

```
#MIPS std default memory map
.eqv text_seg 0x00400000
.eqv data_seg 0x10010000
.eqv heap_seg 0x10040000
.eqv stack_seg 0x7ffffffc
.eqv ktext_seg 0x80000000
.eqv exc_ptr 0x80000180
.eqv kdata_seg 0x90000000
.eqv MMIO_seg 0xffff0000
.eqv memtop_ptr 0xffffffff
#end map
```

MIPS Memory Configuration

- (•) Default
- ( ) Compact, Data at Address 0
- ( ) Compact, Text at Address 0

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0xfffeffff	kernel data segment limit address
0x90000000	.kdata base address
0x8ffffffc	kernel text limit address
0x80000180	exception handler address
0x80000000	kernel space base address
0x80000000	.ktext base address
0x7fffffff	user space high address
0x7fffffff	data segment limit address
0x7ffffffc	stack base address
0x7fffeffc	stack pointer $sp
0x10040000	stack limit address
0x10040000	heap base address
0x10010000	.data base address
0x10008000	global pointer $gp
0x10000000	data segment base address
0x10000000	.extern base address
0x0ffffffc	text limit address
0x00400000	.text base address

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

MARS

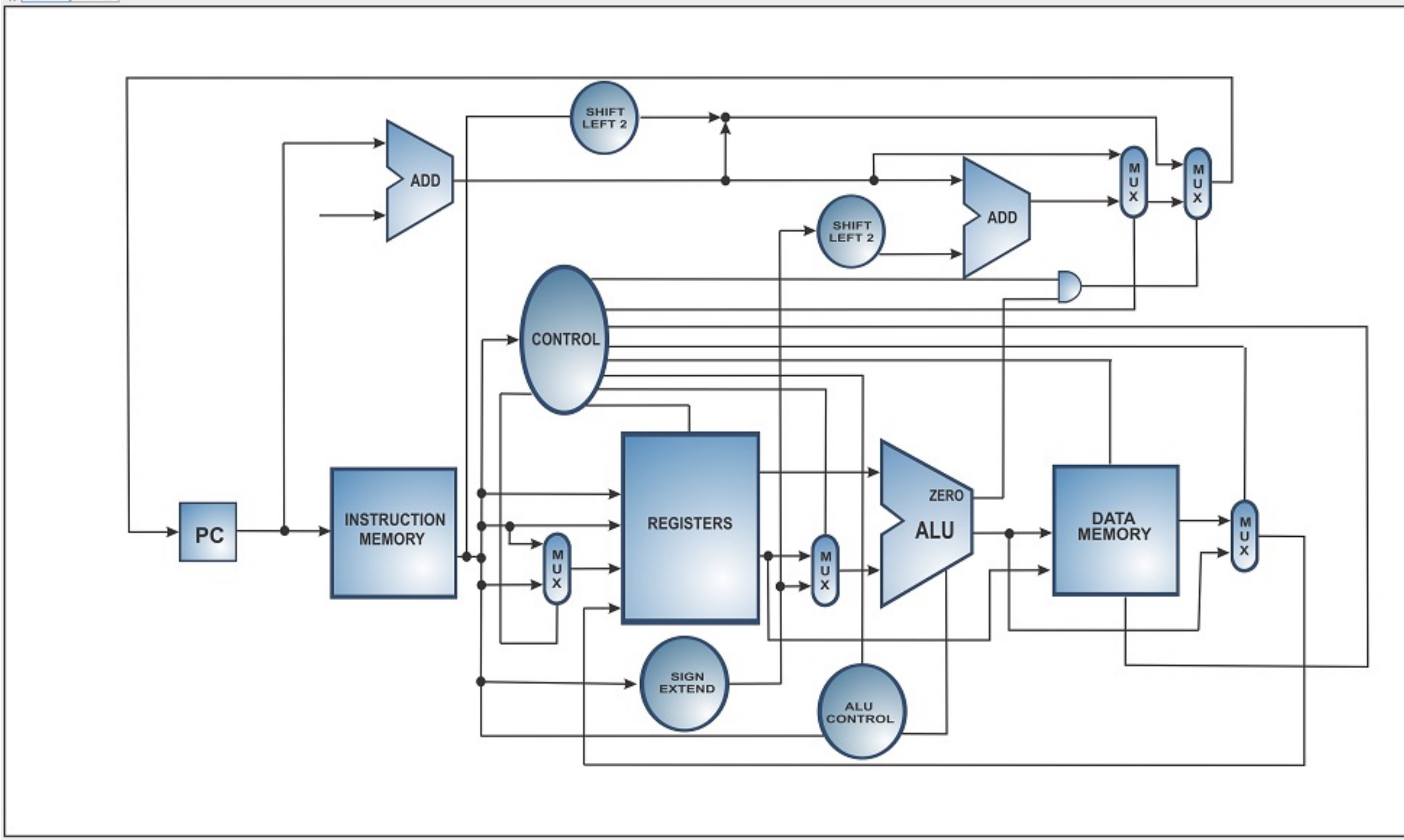
DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

MARS (*MIPS Assembler and Runtime Simulator*)

Tools

MIPS X-Ray – Animation of MIPS Datapath

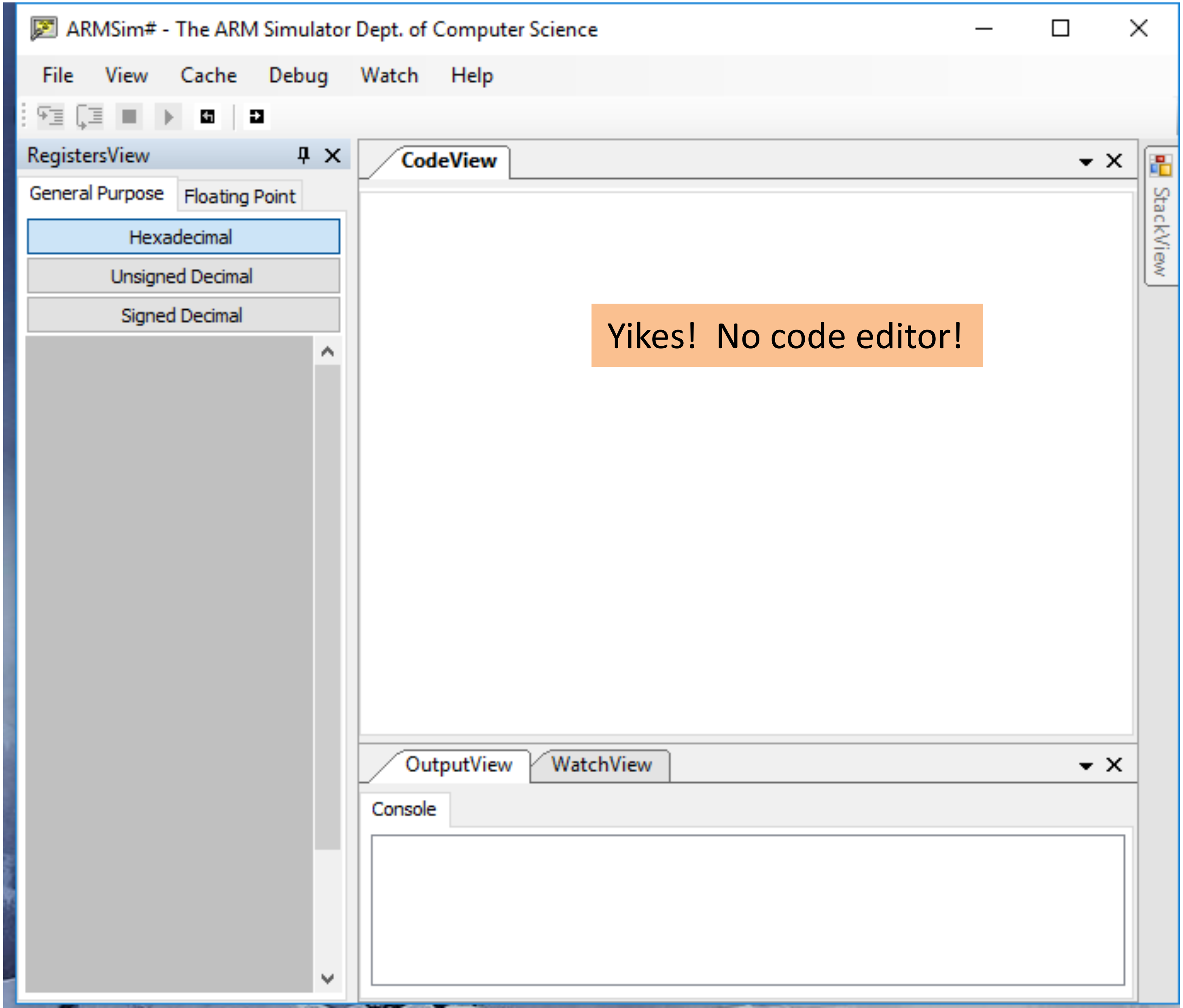
X-Ray



COMP222

ARM Sim

© Jeff Drobman 2020-22



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

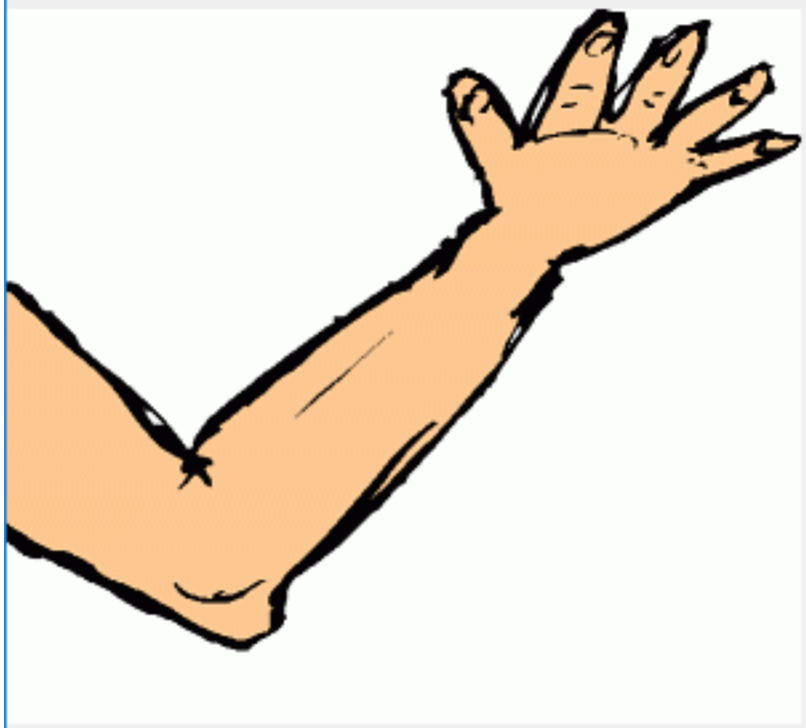
ARM Sim

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

tinyurl.com/armsimcsun

About ARMSim#

ARMSim - the ARM Simulator



ARMSim# Version 2.0.1 (2)

v. 2.0.1

University of Victoria
Produced by:
Dr. Nigel Horspool
Dale Lyons
Dr. Micaela Serra
Bill Bird
Department of Computer Science.

Copyright 2006--2015 University of Victoria.

Simulating ARMv5 instruction architecture with Vector Floating Point support and a Data/Instruction Cache simulation.

ARM Sim

tinyurl.com/armsimcsun

ARMSim# version 2.1 for Windows

The files and installation instructions for use on Windows are provided here.

ARMSim# version 2.1 for Linux

The files and installation instructions for use on Linux are provided here.

ARMSim# version 2.1 for Mac OS X

The files and installation instructions for use on Mac OS X are provided here.

NOT available for Mac!

Mac OS X

NOT available for Mac!

2. Current Distribution Status

ARMSIm# version 2..1 is available for Windows. It has been tested on Windows 8.1.

It has been tested on Ubuntu Linux under Mono. The docking windows feature available on Windows does not work on Linux (due to differences in its support for .NET Forms).

It does not yet work on Mac OS X, apparently due to a difference in the way that scrolling text windows are implemented in Mono on a Mac OS X system.

Mac OS X

Installing ARMSim# on Mac OS X

Choice #1: Run Windows via Dual Boot or Virtualization Software

If you need to run more Windows applications than just ARMSim#, your easiest route is to install the Windows operating system on your Mac computer. Once Windows is installed, you can follow the instructions provided to Windows users for installing ARMSim#. However you do need to own a licensed copy of Windows.

Don't do this!

The possibilities for installing Windows include:

- Use Apple's BootCamp software to configure your Mac computer as a dual-boot machine. Each time you power up the computer, you will have a choice as to whether you want to run the Mac OS X operating system or the Windows operating system.
- Install virtualization software as an application on Mac OS X. The virtualization software will create a virtual machine into which you can install the Windows operating system.

  The possible choices for virtualization software include Parallels (from www.parallels.com), QEMU (from www.qemu.org) and Oracle VirtualBox (from www.virtualbox.org).
- Or both of the above ... after using BootCamp to create a dual boot machine, one can also install Parallels under Mac OS X and have the best of both worlds.

Choice #2: Use Mono on Mac OS X

The open source project, Mono, is an implementation of Microsoft's .NET framework. It can be installed as a Mac OS X application and used to execute the code of the ARMSim# application. **Warning!** Mono does not currently provide all the libraries needed by the docking windows feature

ARMsim 2.1

Introducing ARMSim# Version 2.1

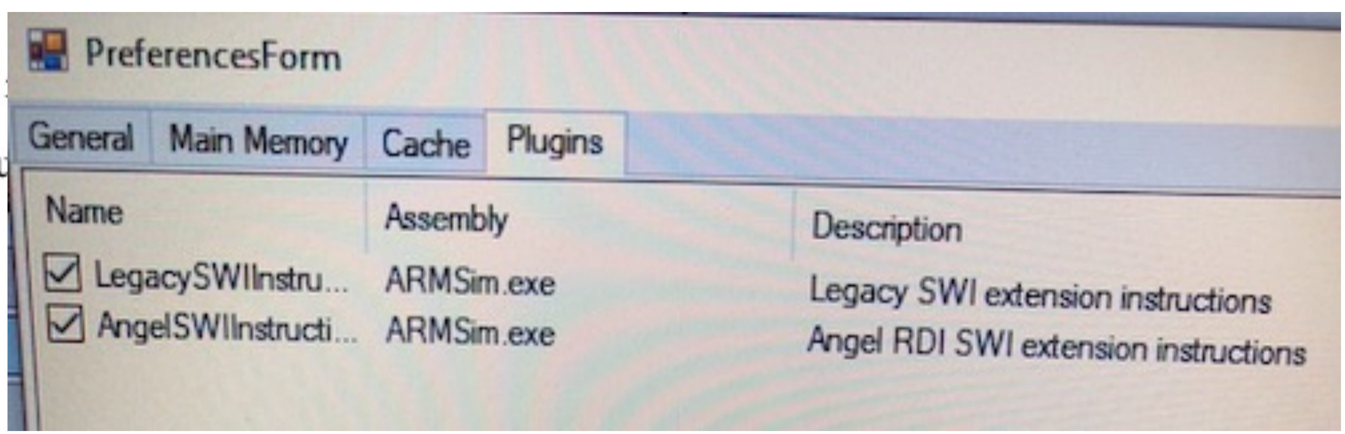
1. What is Different?

Version 2.1 is a major re-design of ARMSim# in three main respects:

1. Instead of parsing and assembling ARM source code itself, ARMSim# now invokes the Gnu Assembler program `as` to perform the task.
2. Instead of using a set of extended SWI instructions based on the ARM RDI family to perform I/O and other system tasks, a new set known as the Angel SWI instructions has been adopted as the default set.
3. The undocumented support for scripting has been replaced by an extended set of command-line options.

Each of these

Some tidying u



ARMsim 2.1

Angel SWI

Adoption of the Angel Extended SWI Instruction Set

The SWI instruction family previously used by ARMSim# was ad hoc and inconsistent because additional features were added piecemeal. This SWI family is *still supported* and we call it the **Legacy SWI** Family.

However, we encourage everyone to switch to the **Angel SWI** Family instead. The reason to do this is that it opens up the possibility of calling functions in the Standard C Library. Many functions in the C Library make calls to the operating system (typically for file and standard I/O access). The version of the C library distributed by Mentor Graphics uses the Angel SWI instruction to request the special services from an operating system.

A disadvantage of the Angel SWI is that the operations are lower level than those provided in the Legacy SWI set. For example, the Legacy SWI provided the ability to input or output decimal numbers, whereas the Angel SWI supports input and ouput of single characters only. As partial compensation, a file containing code to perform some common operations including I/O of numbers with the Angel SWI has been provided. Alternatively, functions such as printf and scanf in the C Library can be invoked.

ARMsim 2.1

Angel SWI

Table 1: Summary of Angel SWI Operations

R0	R1[a]	Description	Operands in Memory (at address provided by R1)
0x01	M	Open a File	Filename address; filename length; file mode
0x02	M	Close a File	File handle
0x05	M	Write to File	File handle; buffer address; number of bytes to write
0x06	M	Read from File	File handle; buffer address; number of bytes to read
0x09	M	Is a TTY?	File handle
0x0A	M	File Seek	File handle; offset from file start
0x0C	M	File Length	File handle
0x0D	M	Temp File Name	Buffer address; unique integer; buffer length
0x0E	M	Remove File	Filename address; filename length
0x0F	M	Rename a File	Filename 1 address; length 1; Filename 2 address; length 2
0x10	–	Execution Time	
0x11	–	Absolute Time	
0x13	–	Get Error Num	
0x16	A	Get Heap Info	
0x18	Code	Exit Program	

a. M indicates the address of the block of operands in memory; A indicates the address of a four word block of memory to receive a result; Code indicates a termination code for the program.

ARMsim 2.1

Angel SWI

```
@    Example of using the Angel SWI operations

        ...                             @ omitted code

        ldr     R1, =OpenParams         @ parameters block for OPEN
        mov     R0, #0x01               @ code number for Open File
        swi     0x123456                @ open a text file for input
        cmp     R0, #0
        blt     OpenError               @ branch if there was an error
        ldr     R1, =ReadParams
        str     R0,[R1]                 @ save the file handle into
                                        @   parameters block for READ
        mov     R0, #0x06               @ code number for Read File
        swi     0x123456                @ read from the text file
        cmp     R0, #0
        bne     ReadError               @ branch if there was an error
```

ARM GNU-A

arm Developer IP PRODUCTS TOOLS AND SOFTWARE ARCHITECTURES INTERNET OF THINGS COMMUNITY SUPPORT DOCUMENTATION DOWNLOADS

GNU-A Downloads

Overview GNU-A GNU-RM Architecture Support Specifications

Downloads

The GNU Toolchain for the Cortex-A Family is a ready-to-use, open source suite of tools for C, C++ and Assembly programming targeting processors from the Arm Cortex-A family and implementing the Arm A-profile architecture.

The toolchain includes the GNU Compiler (GCC) and is available free of charge directly for Windows and Linux operating systems. Follow the links on this page to download the correct version for your development environment.

See the downloaded package's Release Notes (linked from this page) for full installation instructions.

GNU Toolchain for the A-profile Architecture

Version 8.3-2019.03

Released: March 29, 2019

ARM GNU-A

gcc

arm Developer IP PRODUCTS TOOLS AND SOFTWARE ARCHITECTURES INTE

Overview GNU-A GNU-RM Architecture Support Specification

In this release

Windows (i686-mingw32) hosted cross compilers

AArch32 bare-metal target (arm-eabi)

- gcc-arm-8.3-2019.03-i686-mingw32-arm-eabi.tar.xz
- gcc-arm-8.3-2019.03-i686-mingw32-arm-eabi.tar.xz.asc

AArch64 bare-metal target (aarch64-elf)

- gcc-arm-8.3-2019.03-i686-mingw32-aarch64-elf.tar.xz
- gcc-arm-8.3-2019.03-i686-mingw32-aarch64-elf.tar.xz.asc

x86_64 Linux hosted cross compilers

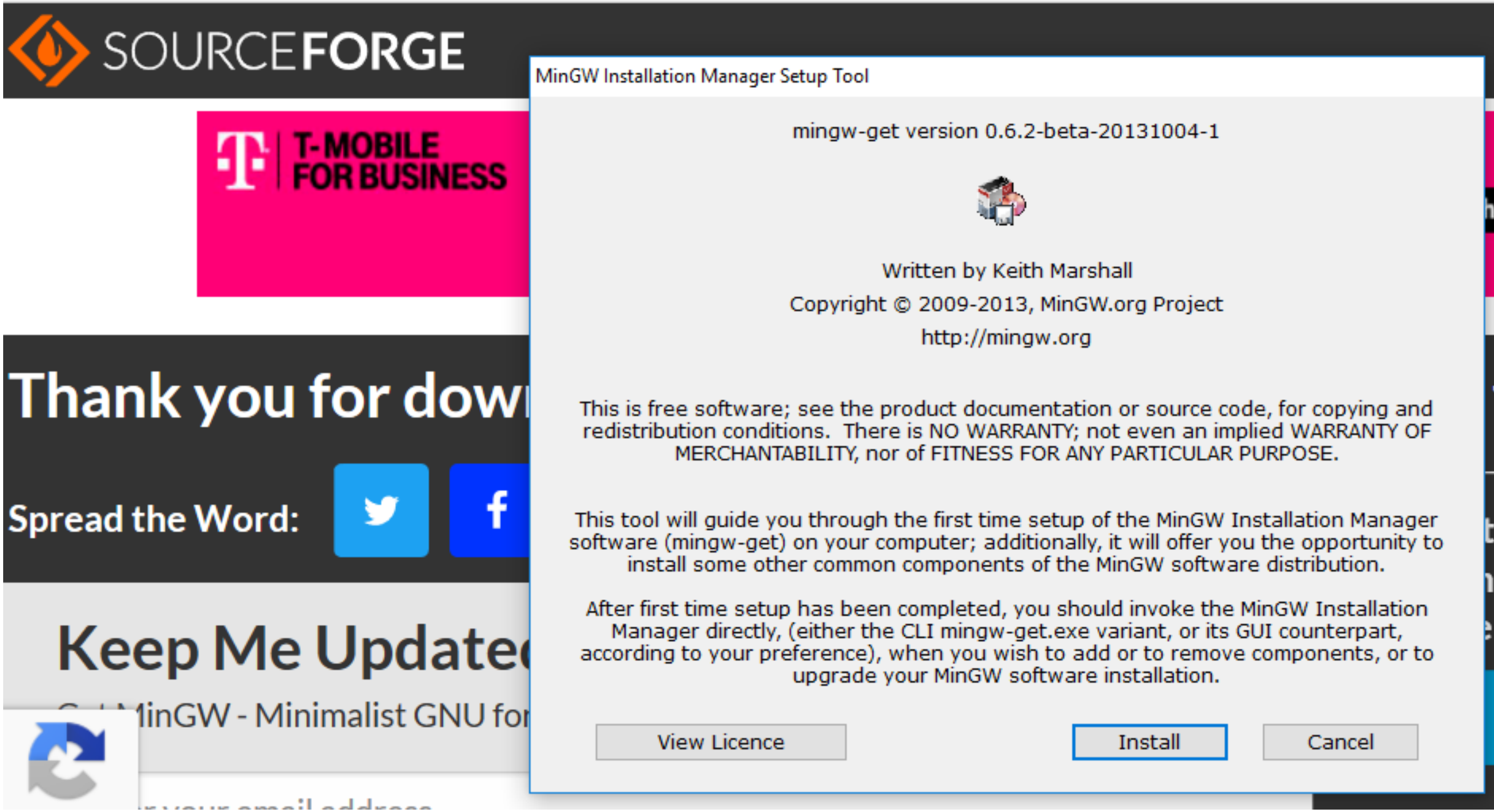
AArch32 bare-metal target (arm-eabi)

- gcc-arm-8.3-2019.03-x86_64-arm-eabi.tar.xz
- gcc-arm-8.3-2019.03-x86_64-arm-eabi.tar.xz.asc

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

GNU - MinGW

DR JEFF SOFTWARE
INDIE APP DEVELOPER



i8085 IDE/Simulator

./ GNUSim8085

A graphical simulator, assembler and debugger for the Intel 8085 microprocessor

Downloads

GNUSim8085 is available in repository of most Linux distributions. If the latest version is not available then you can download source or binaries we provide. Please note that we do not provide binaries for all distributions.

File
Debian/Ubuntu 32 bit (i386) Downloads: 7735
Debian/Ubuntu 64 bit (amd64) Downloads: 11953
Fedora 64 bit (x86_64) Downloads: 8424
Windows 32 bit Downloads: 88480

x86 IDE/Simulators

GEM5

The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture. gem5 is a *community led* project with an open governance model.

gem5 was originally conceived for computer architecture research in academia, but it has grown to be used in computer system design by academia, industry for research, and in teaching.

x86 IDE/Simulators

GEM5

gem5 Home About Documentation Events Contributing Blog Sea

GETTING STARTED WITH GEM5

Getting Started with gem5

First steps

The gem5 simulator is most useful for research when you build new models and new features on top of the current codebase. Thus, the most common way to use gem5 is to download the source and build it yourself.

To download gem5, you can use `git` to checkout to current stable branch. If you're not familiar with version control or git, The git book (available online for free) is a great way to learn more about git and become more comfortable using version control. The canonical version of gem5 is hosted by Google on googlesource.com. However, there is a GitHub mirror as well. It is strongly suggested to use the googlesource version of gem5, and it is required if you want to contribute any changes back to the gem5 mainline.

```
git clone https://gem5.googlesource.com/public/gem5
```

After cloning the source code, you can build gem5 by using `scons`. Building gem5 can take anywhere from a few minutes on a large server to 45 minutes on a laptop. gem5 must be built on a Unix platform. Linux is tested on every commit, and some people have been able to use MacOS as well, though it is not regularly tested. It is strongly suggested to *not* try to compile gem5 when running on a virtual machine. When running with a VM on a laptop gem5 can take over an hour just to compile. The building gem5 provides more details on building gem5 and its dependencies.

Benchmark Index

Section

CPU Metrics

- ❖ MIPS
- ❖ CPI vs. IPC
- ❖ P&H (textbook)

Peak Perf of SMT

IPC = N* (1/CPI)

TYP CPI = 1.3 → 1/1.3 = 0.77

Examples for N at 1 GHz

N = 1 → IPC = 0.77 → MIPS = 770

N = 2 → IPC = 1.54 → MIPS = 1540

N = 4 → IPC = 3.08 → MIPS = 3080

CPU Performance

$$\text{Time} = \text{Seconds/Program} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

Clock rate = 1/Clock cycle time

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

➢ ***Dynamic*** Instruction count

(not Static)

Benchmarks

Hennessy & Patterson

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instructions}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

Time | Size | **CPI** = Fn(ISA) | 1/Freq

Million instructions per second (MIPS):

$$\text{MIPS} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

=MHz/CPI

Amdahl's Law: A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the law of diminishing returns.

U Wisc Slides: Perf

Processor Performance = Time / Program

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) **(CPI)** **(cycle time)**

CPI 1/f

- In the 1980's (decade of pipelining):
  - CPI: 5.0 => 1.15 MIPS
- In the 1990's (decade of superscalar):
  - CPI: 1.15 => 0.5 (best case)
- In the 2000's (decade of multicore):
  - Core CPI unchanged; chip CPI scales with #cores

CPU Performance

CPU performance

- *faster* – *Moore's Law* has guided the shrinkage of transistors which has an attendant increase in frequency. we have seen CPU clock frequency go from 1GHz to up to 5GHz (but mostly 2-3 GHz) today.
- ***performance*** of CPU's is measured in "throughput" = clock period (1/freq) * instructions per cycle per (1/CPI) * number of cores (or execution units, i.e., pipelines), or **Perf = N / (f * CPI)**
- we have seen minor (~5-10%) improvements in CPI over the past 10 years, mostly due to hardware assisted out-of-order and speculative execution. number of ***threads*** has gone up, esp. in wide ***superscalar***.

> - we have essentially reached the end of the line for *scalar single-core* CPU architecture improvements. Processor *frequencies* have been topping out along with the end of Moore's Law transistor shrinkage.
> - so we are now seeing more *parallelism* – in terms of both cores (CPU and GPU) and ***superscalar EU's.***

CPU Performance Factors

❖ factors determining

max **instruction cycle** *frequency*

or minimum cycle *time* per *core* or *pipeline* (if *superscalar*):

1. **transistor switching** frequency (inverse function of feature sizes, e.g., 7-10 nm)
2. single vs. double **clock phases**
3. instruction **cycle times**: determined by slowest pipeline stage
4. gating pipeline stage = longest **logic gate path** in the "ICU" state machine

CPU Performance Factors

❖ Micro Architecture

- ❖ **MT**
- ❖ **Superscalar**
- ❖ **Branch prediction**
- ❖ Instruction **scheduling**
  - ❑ out-of-order
  - ❑ *speculative*
- ❖ **Pipeline** management: operand forwarding

CPU Performance Factors

- **performance issues** that make the CPU run slower than the max frequency:

1. **cache performance**: miss rates, refill rates, line sizes, coherence and locality of reference for both instructions and data
2. **context switch** rates: events such as system calls, traps, exceptions and interrupts.
3. **branch prediction** performance coupled with **branch rates**
4. number of **cores**
5. number of **threads** per core
6. rate of **multi-cycle instructions** such as integer and floating-point multiply, divide, other floating-point operations such as add, subtract.
7. degree of extractable (usable) **parallelism** in the given code determines effective utilization of cores, threads and co-processors (e.g., floating-point units)

CPU Performance

The following table summarizes how these components affect the factors in the CPU performance equation.

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in varied ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

Section

P & H

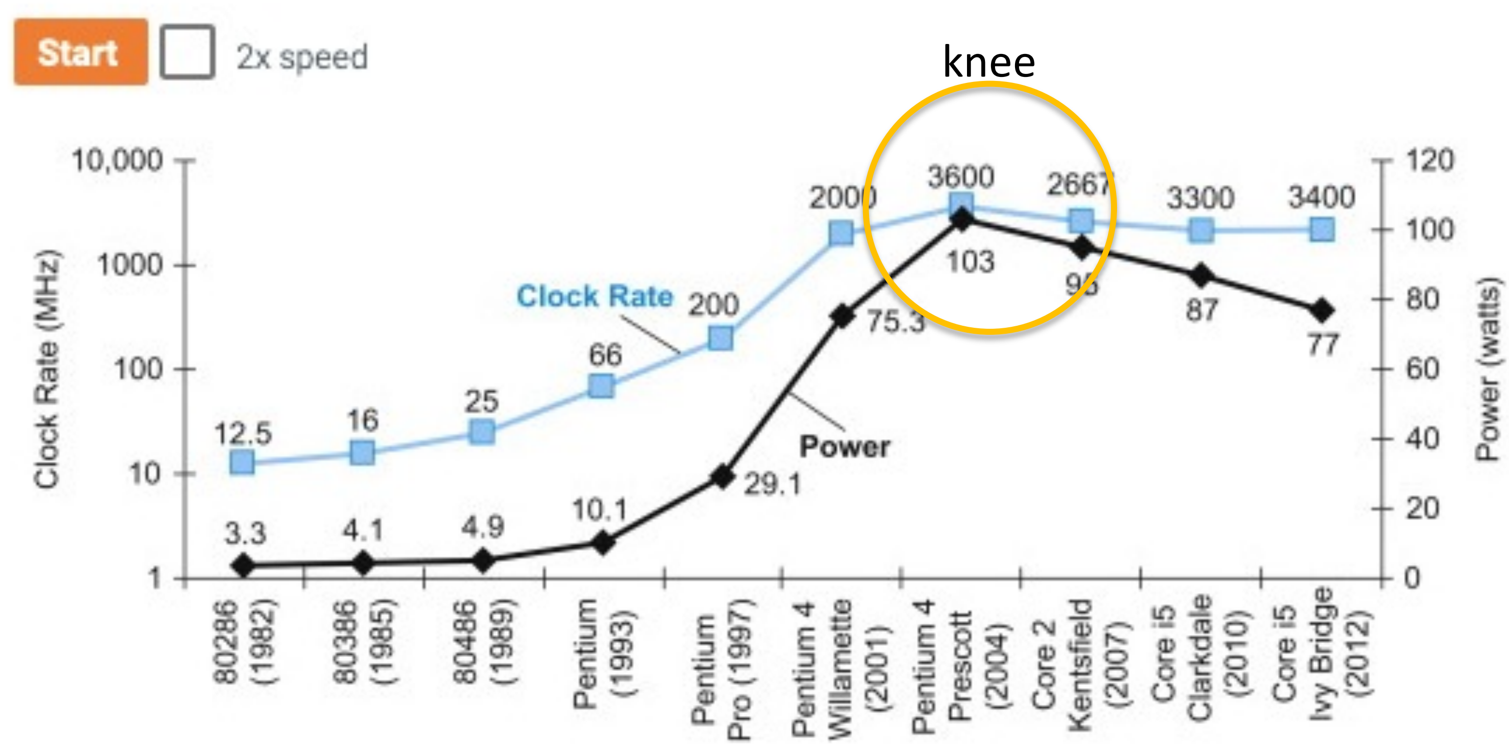
- Sec 6.4
- Sec 6.10
- Sec 6.11

Power Wall

Hennessy & Patterson

1.7 The power wall

PARTICIPATION ACTIVITY | 1.7.1: Clock rate and power for Intel x86 microprocessors over eight generations and 30 years (COD Figure 1.16).

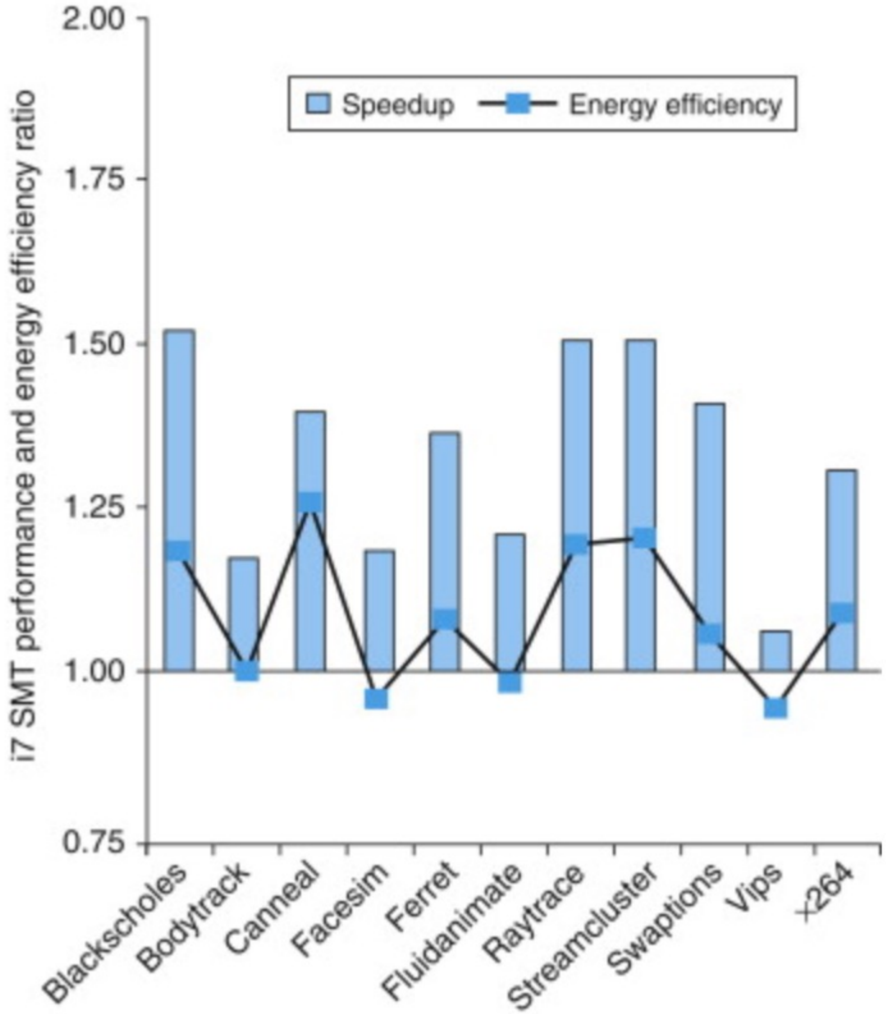


SMT Performance

P&H zyBook

Figure 6.4.2: The speed-up from using multithreading on one core on an i7 processor (COD Figure 6.6).

Processor averages 1.31 for the PARSEC benchmarks (see COD Section 6.9 (Communicating to the outside world: Cluster networking)) and the energy efficiency improvement is 1.0

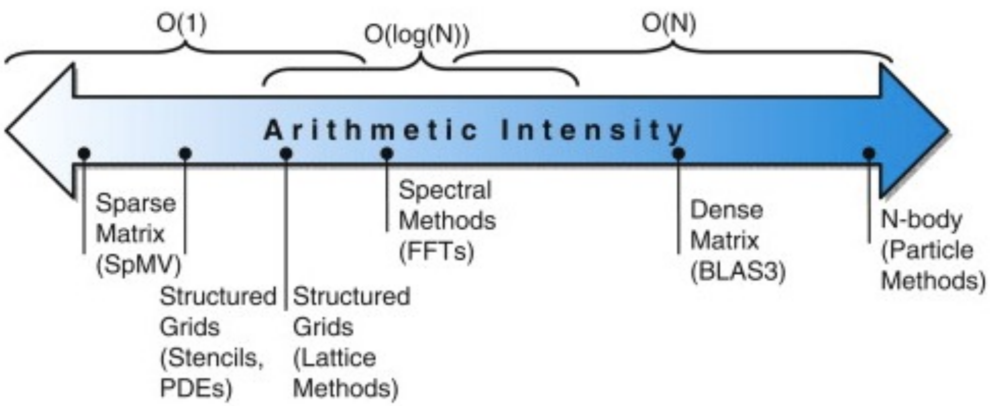


Advanced Perf

P&H Ch 6 (6.10-11)

Figure 6.10.2: Arithmetic intensity, specified as the number of float-point operations to run the program divided by the number of bytes accessed in main memory [Williams, Waterman, and Patterson 2009] (COD Figure 6.17).

Some kernels have an arithmetic intensity that scales with problem size, such as Dense Matrix, but there are many kernels with arithmetic intensities independent of problem size. For kernels in this former case, weak scaling can lead to different results, since it puts much less demand on the memory system.



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Advanced Perf

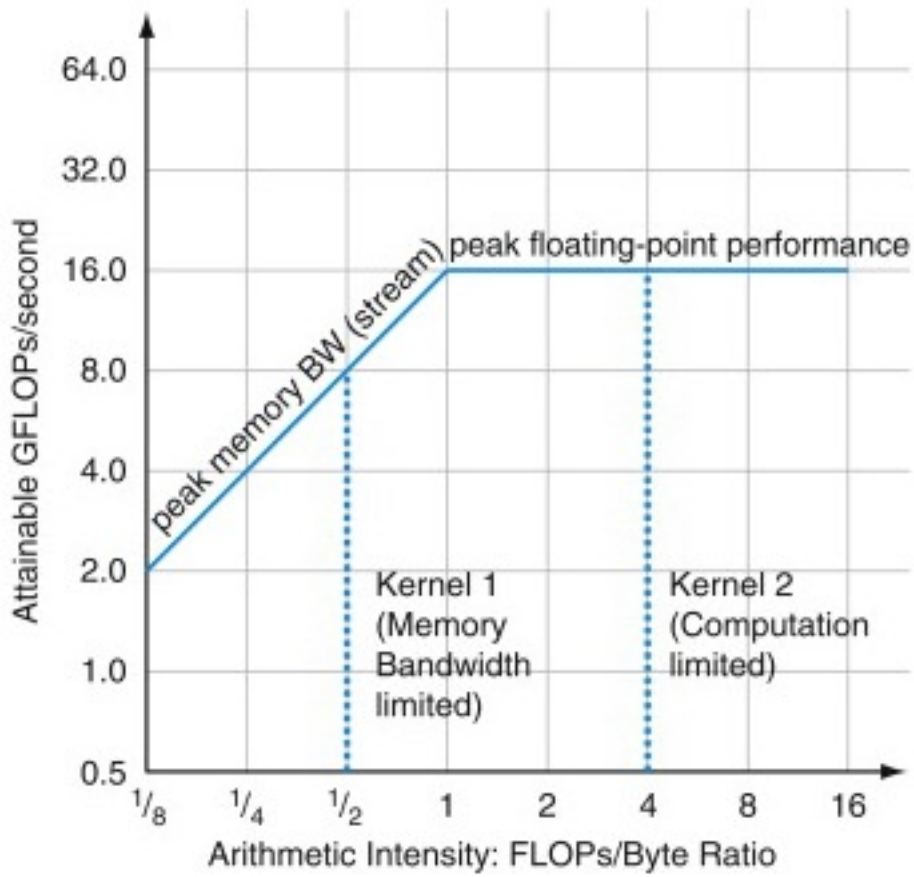
DR JEFF SOFTWARE
INDIE APP DEVELOPER

Floating-point Perf

P&H Ch 6 (6.10-11)

Memory Constraints

Attainable GFLOPs/sec = Min (Peak Memory BW x Arithmetic Intensity, Peak Floating-Point Performance)



Advanced Perf

P&H Ch 6 (6.10-11)

To reduce computational bottlenecks, the following two optimizations can help almost any kernel:

1. *Floating-point operation mix.* Peak floating-point performance for a computer typically requires an equal number of nearly simultaneous additions and multiplications. That balance is necessary either because the computer supports a fused multiply-add instruction (see the Elaboration in COD Section 3.5 (Floating Point)) or because the floating-point unit has an equal number of floating-point adders and floating-point multipliers. The best performance also requires that a significant fraction of the instruction mix is floating- point operations and not integer instructions.

2. *Improve* ***instruction-level parallelism*** *and apply SIMD.* For modern architectures, the highest performance comes when fetching, executing, and committing three to four instructions per clock cycle (see COD Section 4.10 (Parallelism via instructions)). The goal for this step is to improve the code from the compiler to increase ILP. One way is by unrolling loops, as we saw in COD Section 4.12 (Going faster: Instruction-level parallelism and matrix multiply). For the x86 architectures, a single AVX instruction can operate on four double precision operands, so they should be used whenever possible (see COD Sections 3.7 (Real stuff: Streaming SIMD extensions and advanced vector extensions in x86) and 3.8 (Going faster: Subword parallelism and matrix multiply)).

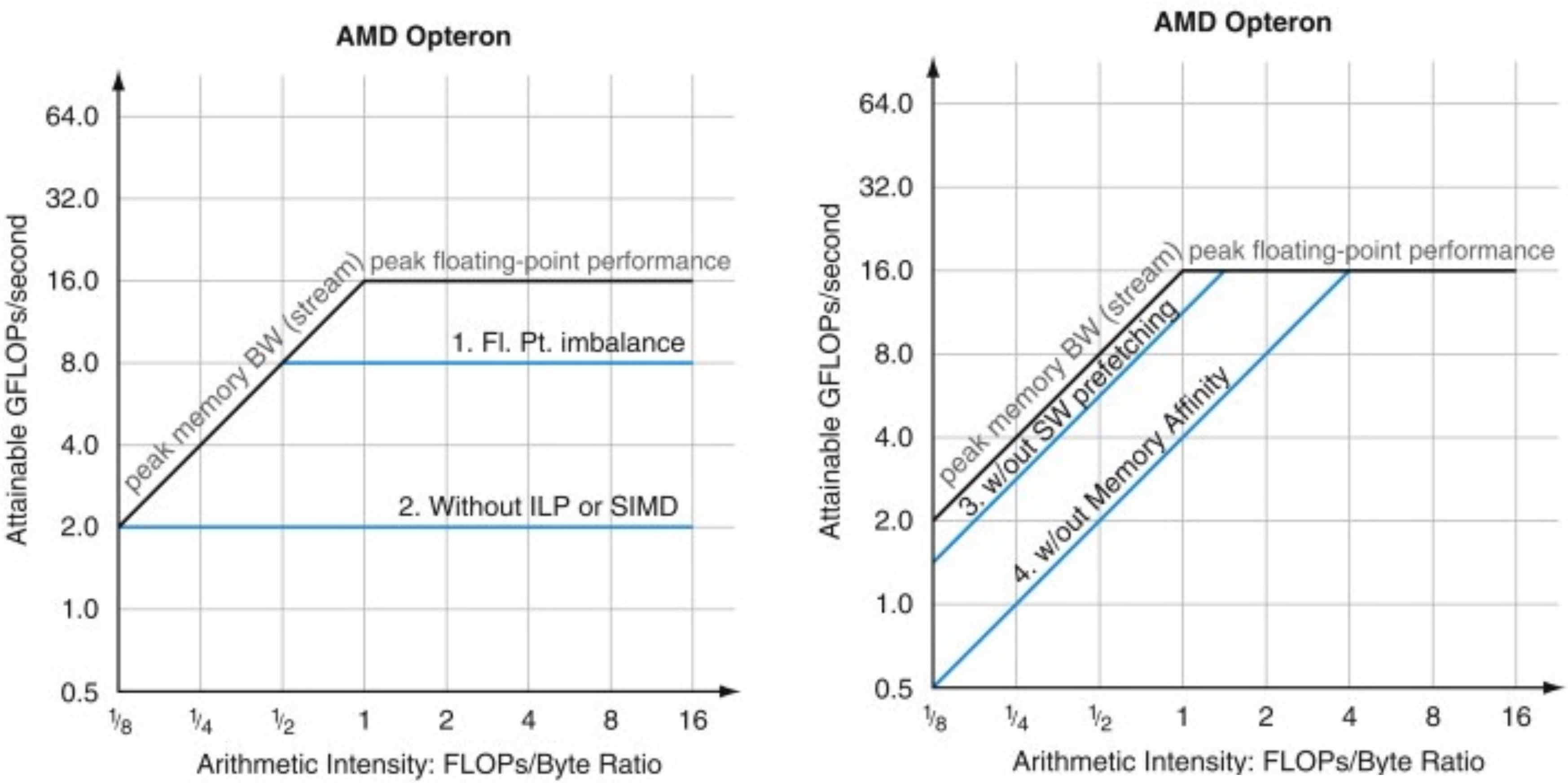
To reduce memory bottlenecks, the following two optimizations can help:

1. *Software prefetching.* Usually the highest performance requires keeping many memory operations in flight, which is easier to do by performing **predicting** accesses via software prefetch instructions rather than waiting until the data is required by the computation.
2. *Memory affinity.* Microprocessors today include a memory controller on the same chip with the microprocessor, which improves performance of the **memory hierarchy**. If the system has multiple chips, this means that some addresses go to the DRAM that is local to one chip, and the rest require accesses over the chip interconnect to access the DRAM that is local to another chip. This split results in non-uniform memory accesses, which we described in COD Section 6.5 (Multicore and other shared memory multiprocessors). Accessing memory through another chip lowers performance. This second optimization tries to allocate data and the threads tasked to operate on that data to the same memory-processor pair, so that the processors rarely have to access the memory of the other chips.

Advanced Perf

P&H Ch 6 (6.10-11)

AMD Opteron

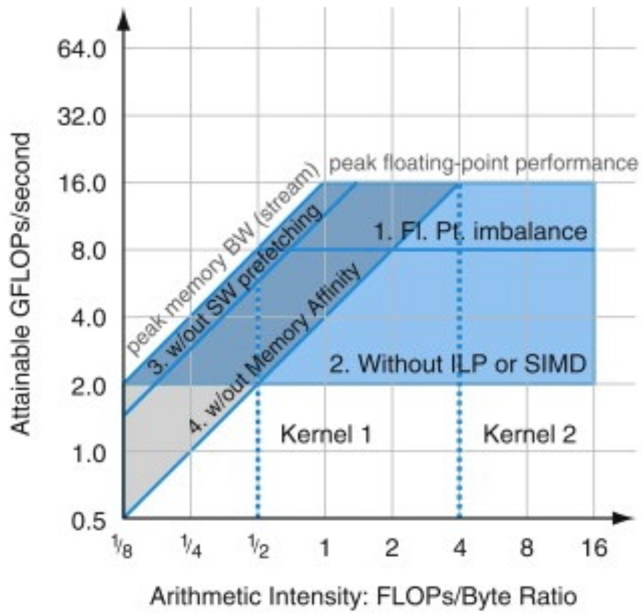


Advanced Perf

P&H Ch 6 (6.10-11)

Figure 6.10.6: Roofline model with ceiling, overlapping areas shaded, and the two kernels from COD Figure 6.18 (Roofline Model ...) (COD Figure 6.21).

Kernels whose arithmetic intensity land in the blue trapezoid on the right should focus on computation optimizations, and kernels whose arithmetic intensity land in the gray triangle in the lower left should focus on memory bandwidth optimizations. Those that land in the blue-gray parallelogram in the middle need to worry about both. As Kernel 1 falls in the parallelogram in the middle, try optimizing ILP and SIMD, memory affinity, and software prefetching. Kernel 2 falls in the trapezoid on the right, so try optimizing ILP and SIMD and the balance of floating-point operations.



Advanced Perf

P&H Ch 6 (6.10-11)

Figure 6.11.1: Intel Core i7-960, NVIDIA GTX 280, and GTX 480 specifications (COD Figure 6.22).

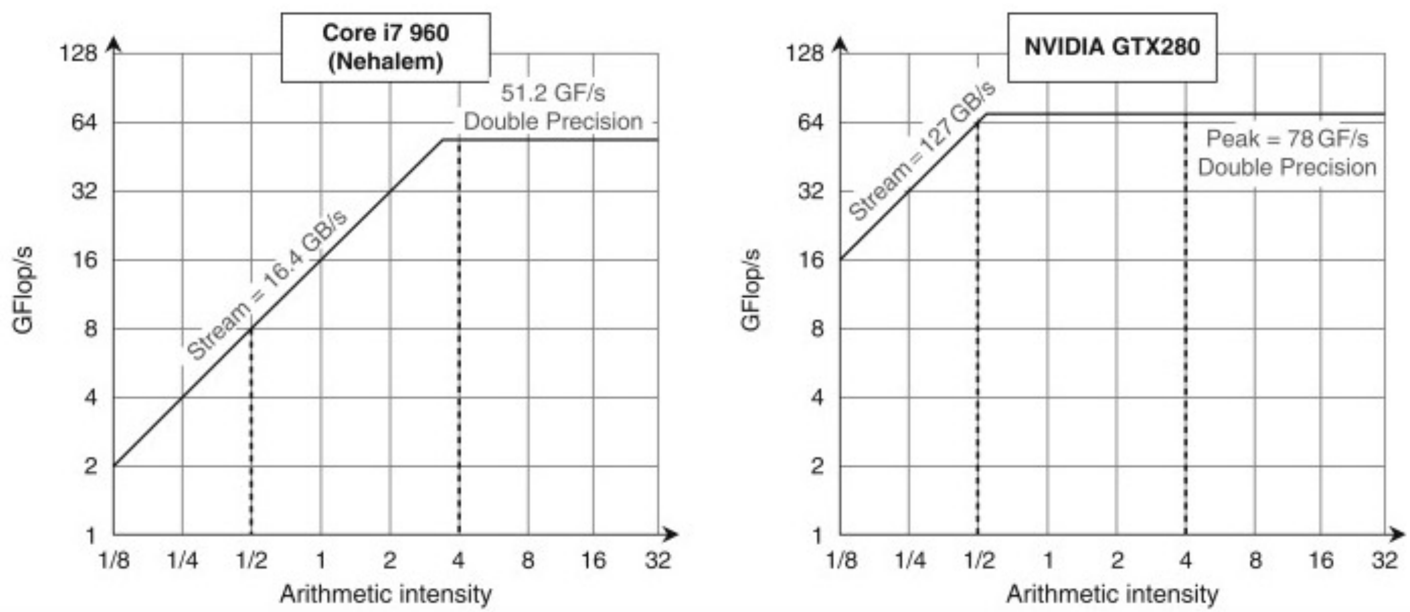
The rightmost columns show the ratios of the Tesla GTX 280 and the Fermi GTX 480 to Core i7. Although the case study is between the Tesla 280 and i7, we include the Fermi 480 to show its relationship to the Tesla 280 since it is described in this chapter. Note that these memory bandwidths are higher than in the figure below because these are DRAM pin bandwidths and those in the figure below are at the processors as measured by a benchmark program. (From Table 2 in Lee et al. [2010].)

	Core i7-960	GTX 280	GTX 480	Ratio 280/i7	Ratio 480/i7
Number of processing elements (cores or SMs)	4	30	15	7.5	3.8
Clock frequency (GHz)	3.2	1.3	1.4	0.41	0.44
Die size	263	576	520	2.2	2.0
Technology	Intel 45 nm	TSMC 65 nm	TSMC 40 nm	1.6	1.0
Power (chip, not module)	130	130	167	1.0	1.3
Transistors	700 M	1400 M	3030 M	2.0	4.4
Memory brandwith (GBytes/sec)	32	141	177	4.4	5.5
Single-precision SIMD width	4	8	32	2.0	8.0
Double-precision SIMD width	2	1	16	0.5	8.0
Peak Single-precision scalar FLOPS (GFLOP/sec)	26	117	63	4.6	2.5
Peak Single-precision SIMD FLOPS (GFLOP/Sec)	102	311 to 933	515 or 1344	3.0–9.1	6.6–13.1
(SP 1 add or multiply)	N.A.	(311)	(515)	(3.0)	(6.6)
(SP 1 instruction fused multiply-adds)	N.A.	(622)	(1344)	(6.1)	(13.1)
(Rare SP dual issue fused multiply-add and multiply)	N.A.	(933)	N.A.	(9.1)	–
Peak double-precision SIMD FLOPS (GFLOP/sec)	51	78	515	1.5	10.1

Advanced Perf

Figure 6.11.2: Roofline model [Williams, Waterman, and Patterson 2009] (COD Figure 6.23).

These rooflines show double-precision floating-point performance in the top row and single-precision performance in the bottom row. (The DP FP performance ceiling is also in the bottom row to give perspective.) The Core i7 960 on the left has a peak DP FP performance of 51.2 GFLOP/sec, a SP FP peak of 102.4 GFLOP/sec, and a peak memory bandwidth of 16.4 GBytes/sec. The NVIDIA GTX 280 has a DP FP peak of 78 GFLOP/sec, SP FP peak of 624 GFLOP/sec, and 127 GBytes/sec of memory bandwidth. The dashed vertical line on the left represents an arithmetic intensity of 0.5 FLOP/byte. It is limited by memory bandwidth to no more than 8 DP GFLOP/sec or 8 SP GFLOP/sec on the Core i7. The dashed vertical line to the right has an arithmetic intensity of 4 FLOP/byte. It is limited only computationally to 51.2 DP GFLOP/sec and 102.4 SP GFLOP/sec on the Core i7 and 78 DP GFLOP/sec and 624 SP GFLOP /sec on the GTX 280. To hit the highest computation rate on the Core i7 you need to use all 4 cores and SSE instructions with an equal number of multiplies and adds. For the GTX 280, you need to use fused multiply-add instructions on all multithreaded SIMD processors.



Advanced Perf

P&H Ch 6 (6.10-11)

Figure 6.11.3: Raw and relative performance measured for the two platforms (COD Figure 6.24).

In this study, SAXPY is just used as a measure of memory bandwidth, so the right unit is GBytes/sec and not GFLOP/sec. (Based on Table 3 in [Lee et al., 2010].)

Kernel	Units	Core i7-960	GTX 280	GTX 280/ i7-960
SGEMM	GFLOP/sec	94	364	3.9
MC	Billion paths/sec	0.8	1.4	1.8
Conv	Million pixels/sec	1250	3500	2.8
FFT	GFLOP/sec	71.4	213	3.0
SAXPY	GBytes/sec	16.8	88.8	5.3
LBM	Million lookups/sec	85	426	5.0
Solv	Frames/sec	103	52	0.5
SpMV	GFLOP/sec	4.9	9.1	1.9
GJK	Frames/sec	67	1020	15.2
Sort	Million elements/sec	250	198	0.8
RC	Frames/sec	5	8.1	1.6
Search	Million queries/sec	50	90	1.8
Hist	Million pixels/sec	1517	2583	1.7
Bilat	Million pixels/sec	83	475	5.7

Section

Std Benchmarks

Single Core

Benchmarks

- Perf metrics
  - Functions/steps/loops per second
    - ***Big*** is better
  - Seconds per Function/step/loop
    - ***Small*** is better
- Old std benchmarks
  - Whetstones
    - ***Floating-point***
  - Dhrystones
    - ***Integer***

Whetstones

FP

Whetstone (benchmark)

From Wikipedia, the free encyclopedia

The **Whetstone benchmark** is a synthetic benchmark for evaluating the performance of computers.[1] It was first written in Algol 60 in 1972 at the Technical Support Unit of the Department of Trade and Industry (later part of the Central Computer and Telecommunications Agency) in the United Kingdom). It was derived from statistics on program behaviour gathered on the KDF9 computer at NPL National Physical Laboratory, using a modified version of its Whetstone ALGOL 60 compiler. The workload on the machine was represented as a set of frequencies of execution of the 124 instructions of the Whetstone Code. The Whetstone Compiler was built at the Atomic Power Division of the English Electric Company in Whetstone, Leicestershire, England,[2] hence its name. Dr. B.A. Wichman at NPL produced a set of 42 simple ALGOL 60 statements, which in a suitable combination matched the execution statistics.

By strict definition, the term *whetsone* refers to a sharpening stone utilized to hone a sharp edge on a steel utensil such as a knife; the obvious reference here is to improve the quality or performance of code by honing its characteristics against the benchmark.

To make a more practical benchmark Harold Curnow of TSU wrote a program incorporating the 42 statements. This program worked in its ALGOL 60 version, but when translated into FORTRAN it was not executed correctly by the IBM optimizing compiler. Calculations whose results were not output were omitted. He then produced a set of program fragments which were more like real code and which collectively matched the original 124 Whetstone instructions. Timing this program gave a measure of the machine's speed in thousands of Whetstone instructions per second (kWIPS). The Fortran version became the first general purpose benchmark that set industry standards of computer system performance. Further development was carried out by Roy Longbottom, also of TSU/CCTA, who became the official design authority. The Algol 60 program ran under the Whetstone compiler in July 2010, for the first time since the last KDF9 was shut down in 1980, but now executed by a KDF9 emulator.[3] Following increased computer speeds, performance measurement was changed to Millions of Whetstone Instructions Per Second (MWIPS).

Source code and pre-compiled versions for PCs in C/C++, Basic, Visual Basic, Fortran and Java are available.[4][5]

The Whetstone benchmark primarily measures the floating-point arithmetic performance. A similar benchmark for integer and string operations is the Dhrystone.

whet·stone | ˈ(h)wetˌstōn |

noun

a fine-grained stone used for sharpening cutting tools.

Dhrystones

Integer

Dhrystone

From Wikipedia, the free encyclopedia

Dhrystone is a synthetic computing benchmark program developed in 1984 by Reinhold P. Weicker intended to be representative of system (integer) programming. The Dhrystone grew to become representative of general processor (CPU) performance. The name "Dhrystone" is a pun on a different benchmark algorithm called Whetstone.[1]

With Dhrystone, Weicker gathered meta-data from a broad range of software, including programs written in FORTRAN, PL/1, SAL, ALGOL 68, and Pascal. He then characterized these programs in terms of various common constructs: procedure calls, pointer indirections, assignments, etc. From this he wrote the Dhrystone benchmark to correspond to a representative mix. Dhrystone was published in Ada, with the C version for Unix developed by Rick Richardson ("version 1.1") greatly contributing to its popularity.

Dhrystone vs. Whetstone [edit]

The Dhrystone benchmark contains no floating point operations, thus the name is a pun on the then-popular Whetstone benchmark for floating point operations. The output from the benchmark is the number of Dhrystones per second (the number of iterations of the main code loop per second).

Both Whetstone and Dhrystone are *synthetic* benchmarks, meaning that they are simple programs that are carefully designed to statistically mimic the processor usage of some common set of programs. Whetstone, developed in 1972, originally strove to mimic typical Algol 60 programs based on measurements from 1970, but eventually became most popular in its Fortran version, reflecting the highly numerical orientation of computing in the 1960s.

Dhrystones

Integer

DEC VAX 11/780 = 1 MIPS

Results [edit]

Dhrystone may represent a result more meaningfully than MIPS (million instructions per second) because instruction count comparisons between different instruction sets (e.g. RISC vs. CISC) can confound simple comparisons. For example, the same high-level task may require many more instructions on a RISC machine, but might execute faster than a single CISC instruction. Thus, the Dhrystone score counts only the number of program iteration completions per second, allowing individual machines to perform this calculation in a machine-specific way. Another common representation of the Dhrystone benchmark is the **DMIPS** (Dhrystone MIPS) obtained when the Dhrystone score is divided by 1757 (the number of Dhrystones per second obtained on the VAX 11/780, nominally a 1 MIPS machine).

Another way to represent results is in DMIPS/MHz, where DMIPS result is further divided by CPU frequency, to allow for easier comparison of CPUs running at different clock rates.

Shortcomings [edit]

Using Dhrystone as a benchmark has pitfalls:

- It features unusual code that is not usually representative of real-life programs.[2]
- It is susceptible to compiler optimizations. For example, it does a lot of string copying in an attempt to measure string copying performance. However, the strings in Dhrystone are of known constant length and their starts are aligned on natural boundaries, two characteristics usually absent from real programs. Therefore, an optimizer can replace a string copy with a sequence of word moves without any loops, which will be much faster. This optimization consequently overstates system performance, sometimes by more than 30%.[3]
- Dhrystone's small code size may fit in the instruction cache of a modern CPU, so that instruction fetch performance is not rigorously tested.[2]

Benchmarks

Snapdragon 636

Benchmarks

3DMark - Ice Storm Unlimited Physics 1280x720 offscreen

min: 15766 avg: 17134.3 median: **17002** (20%) max: 19365 Points

11 benchmarks and specifications | Show comparison chart

3DMark - Sling Shot Extreme (ES 3.1) Unlimited Physics 2560x1440

min: 2258 avg: 2375.6 median: **2364** (52%) max: 2644 Points

10 benchmarks and specifications | Show comparison chart

3DMark - Sling Shot OpenGL ES 3.0 Unlimited Physics 2560x1440

min: 2343 avg: 2395.9 median: **2365** (50%) max: 2683 Points

10 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Stream

2016 Points (16%)

1 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Memory

2801 Points (25%)

1 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Floating Point

8155 Points (23%)

1 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Integer

4063 Points (13%)

1 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Total Score

5063 Points (19%)

1 benchmarks and specifications | Show comparison chart

MediaTek Helio P60

Benchmarks

3DMark - Ice Storm Unlimited Physics 1280x720 offscreen

min: 18065 avg: 22578.4 median: **23697** (27%) max: 24828 Points

5 benchmarks and specifications | Show comparison chart

3DMark - Sling Shot Extreme (ES 3.1) Unlimited Physics 2560x1440

min: 2013 avg: 2519.8 median: **2616** (58%) max: 2918 Points

5 benchmarks and specifications | Show comparison chart

3DMark - Sling Shot OpenGL ES 3.0 Unlimited Physics 2560x1440

min: 2388 avg: 2710.5 median: **2765** (59%) max: 2924 Points

4 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Stream

min: 2154 avg: 2270 median: **2270** (18%) max: 2386 Points

2 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Memory

min: 3119 avg: 3494 median: **3494** (32%) max: 3869 Points

2 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Floating Point

min: 9919 avg: 10532.5 median: **10532** (30%) max: 11146 Points

2 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Integer

min: 4782 avg: 5083.5 median: **5083** (16%) max: 5385 Points

2 benchmarks and specifications | Show comparison chart

Geekbench 2 - 32 Bit - Total Score

min: 5984 avg: 6391 median: **6391** (24%) max: 6798 Points

2 benchmarks and specifications | Show comparison chart

Std Benchmarks

Hennessy & Patterson

Table 1.9.1 SPECINTC2006 benchmarks running on a 2.66GHz Intel Core i7 920 (COD Figure 1.18).

Execution time is the product of the three factors in this table: instruction count in billions, clocks per instruction (CPI), and clock cycle time in nanoseconds. SPECratio is simply the reference time, which is supplied by SPEC, divided by the measured execution time. The single number quoted as SPECINTC2006 is the geometric mean of the SPECratios.

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9

Performance

Hennessy & Patterson

1.12: Historical perspective and reading

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price/ performance vs. UNIVAC	Adjusted price (2007 $)	Adjusted price/ performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$7,670,724	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$6,018,798	319
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$94,685	13,367
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$13,509,798	47,127
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$6,859	134,208
1991	HP 9000/model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$11,807	16,241,889
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$6,211	247,021,234
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$2,009	11,451,750,000
2007	AMD Barcelona PC (2.5 GHz)	2	250	20,000,000,000	2,097,152	$800	12,500,000,000	$800	95,884,051,042

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP222

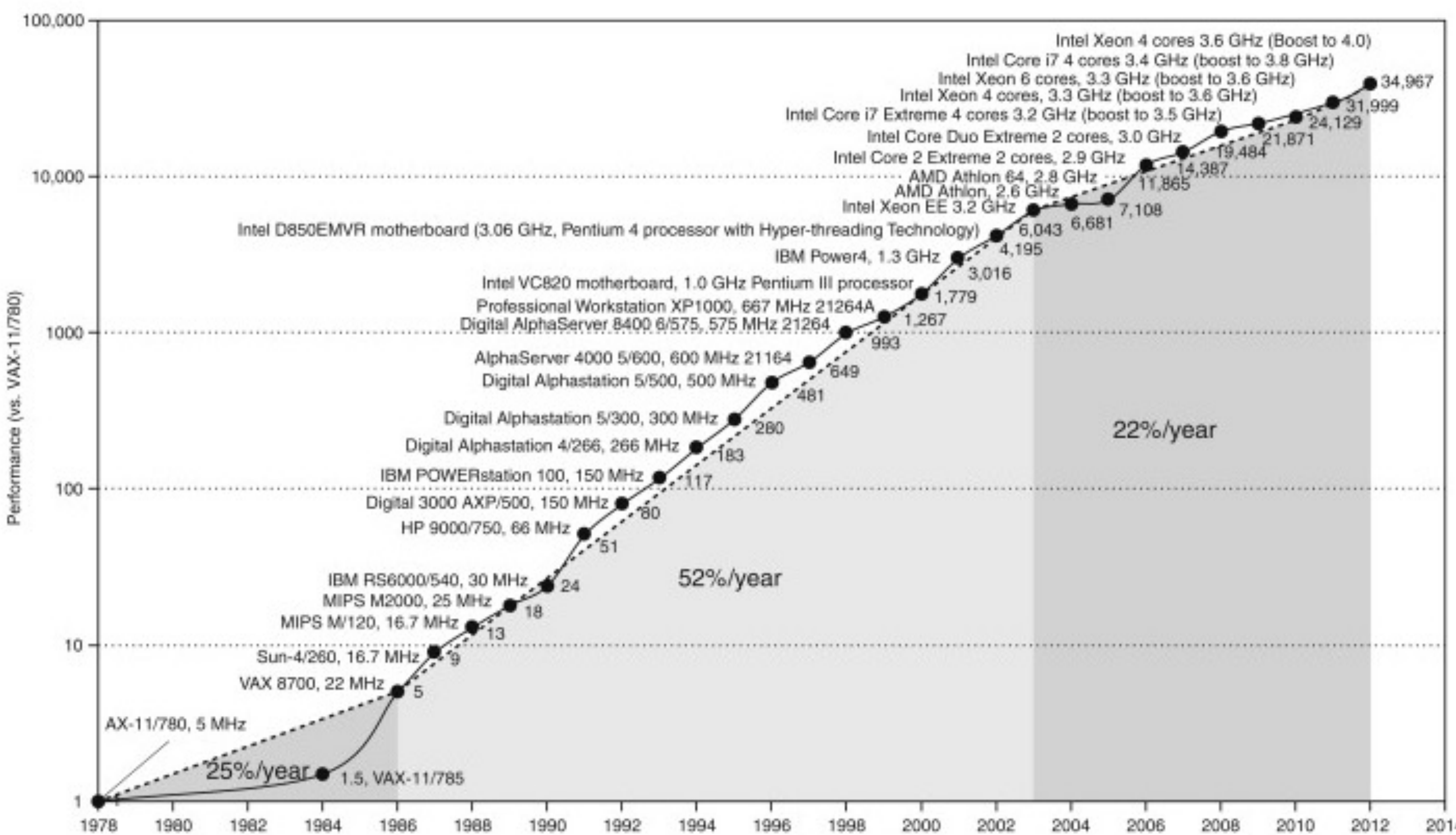
Performance

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Hennessy & Patterson

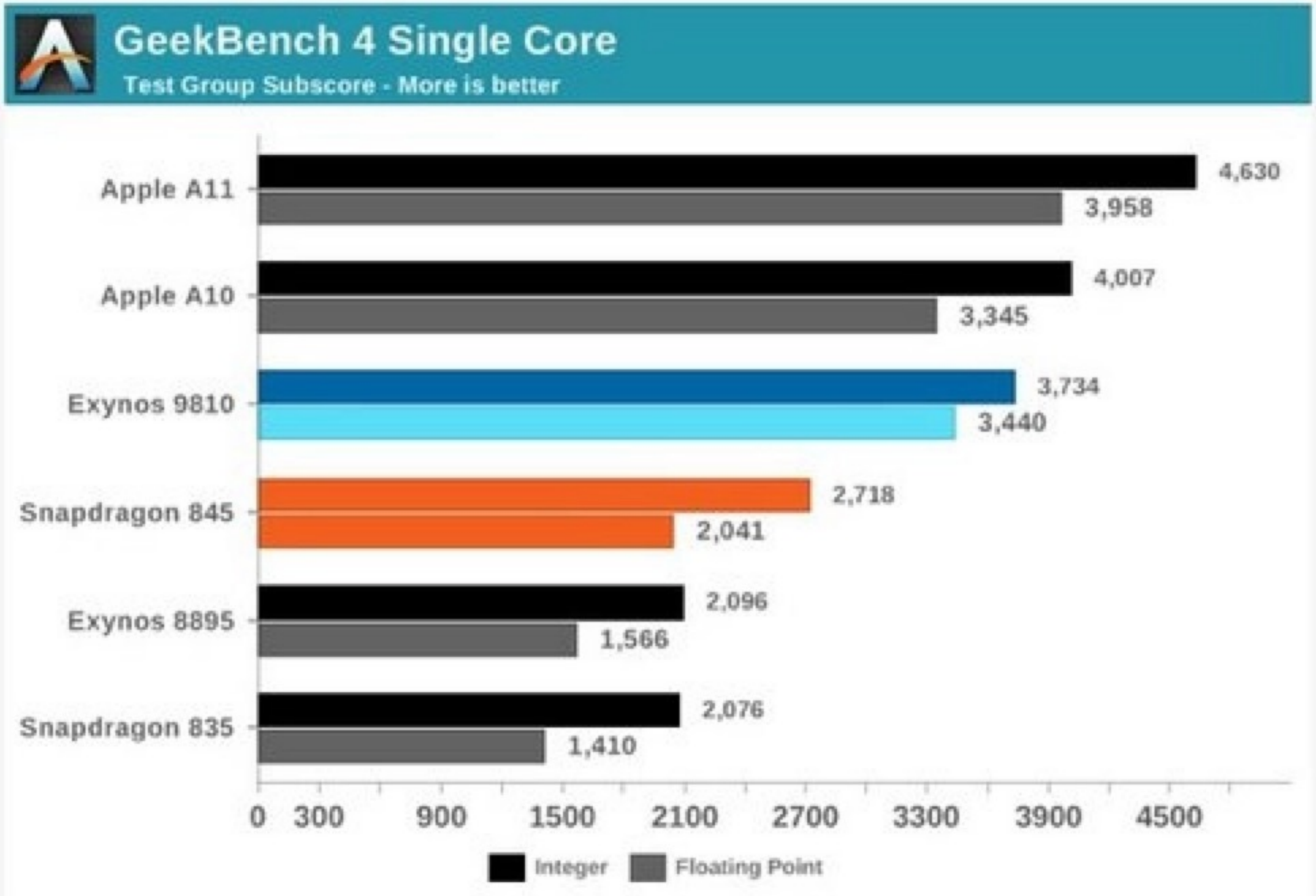
Figure 1.8.1: Growth in processor performance since the mid-1980s (COD Figure 1.17).

This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks (see COD Section 1.10 (Fallacies and pitfalls)). Prior to the mid-1980s, processor performance growth was largely technology-driven and averaged about 25% per year. The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. The higher annual performance improvement of 52% since the mid-1980s meant performance was about a factor of seven higher in 2002 than it would have been had it stayed at 25%. Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 22% per year.



Amey Apte · Follow
Knows some things about them.
Quora

Benchmark



In the graphics dept, the SD845 maintains a slight edge, whereas the results of the PCMark tests are an anomaly most likely caused due to unoptimized software before hitting retail distribution.

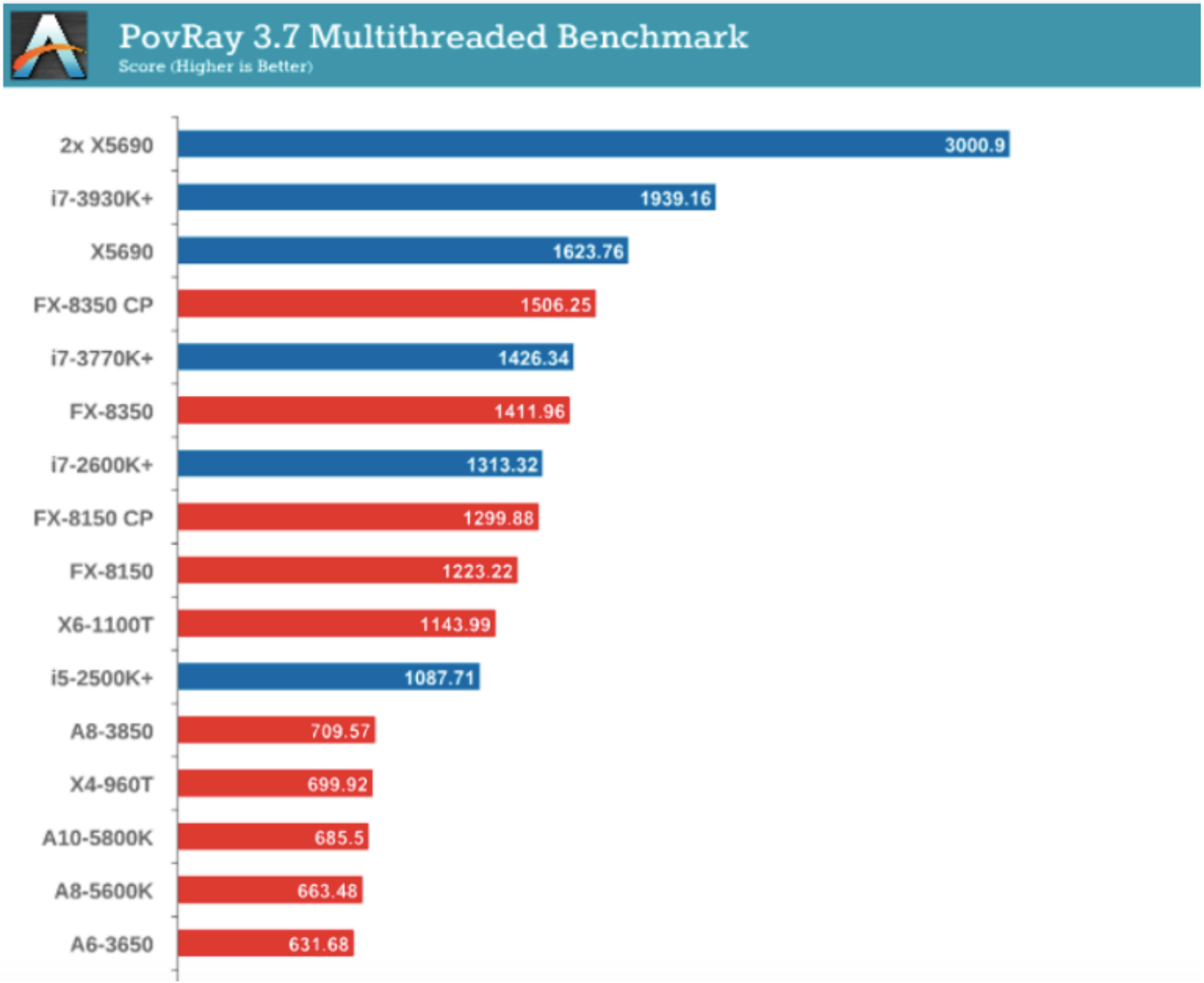
Section

Multi-threading

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

MT

DR JEFF SOFTWARE
INDIE APP DEVELOPER

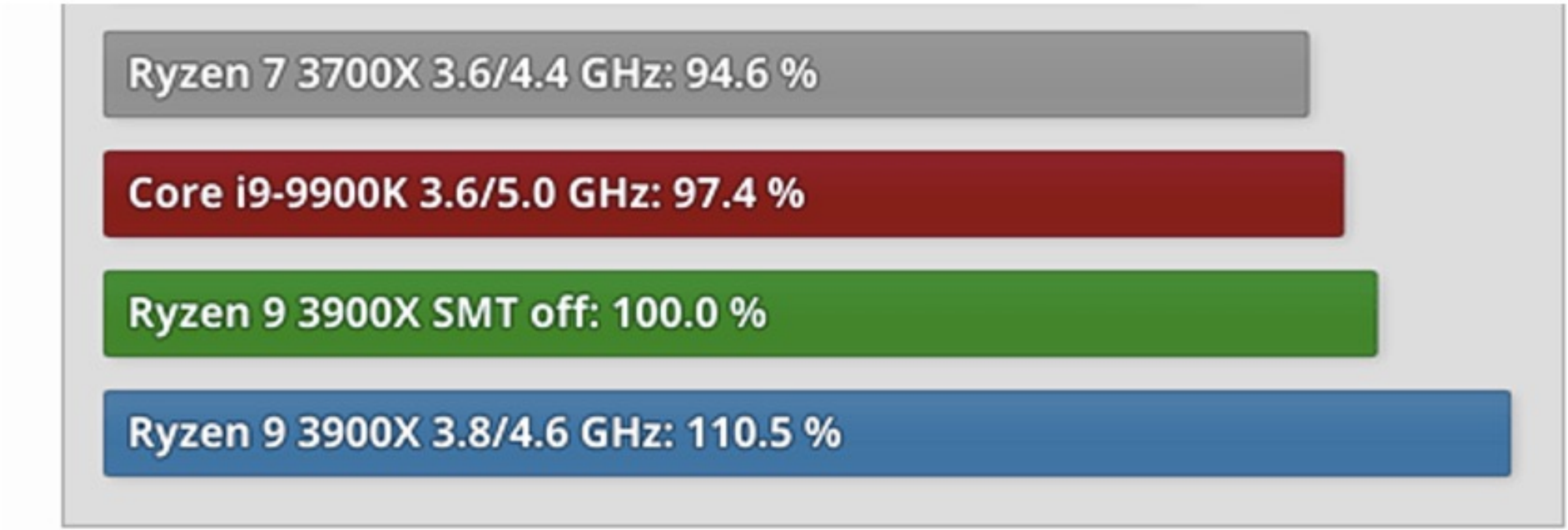


CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Misc Benchmarks

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Multi-Threading

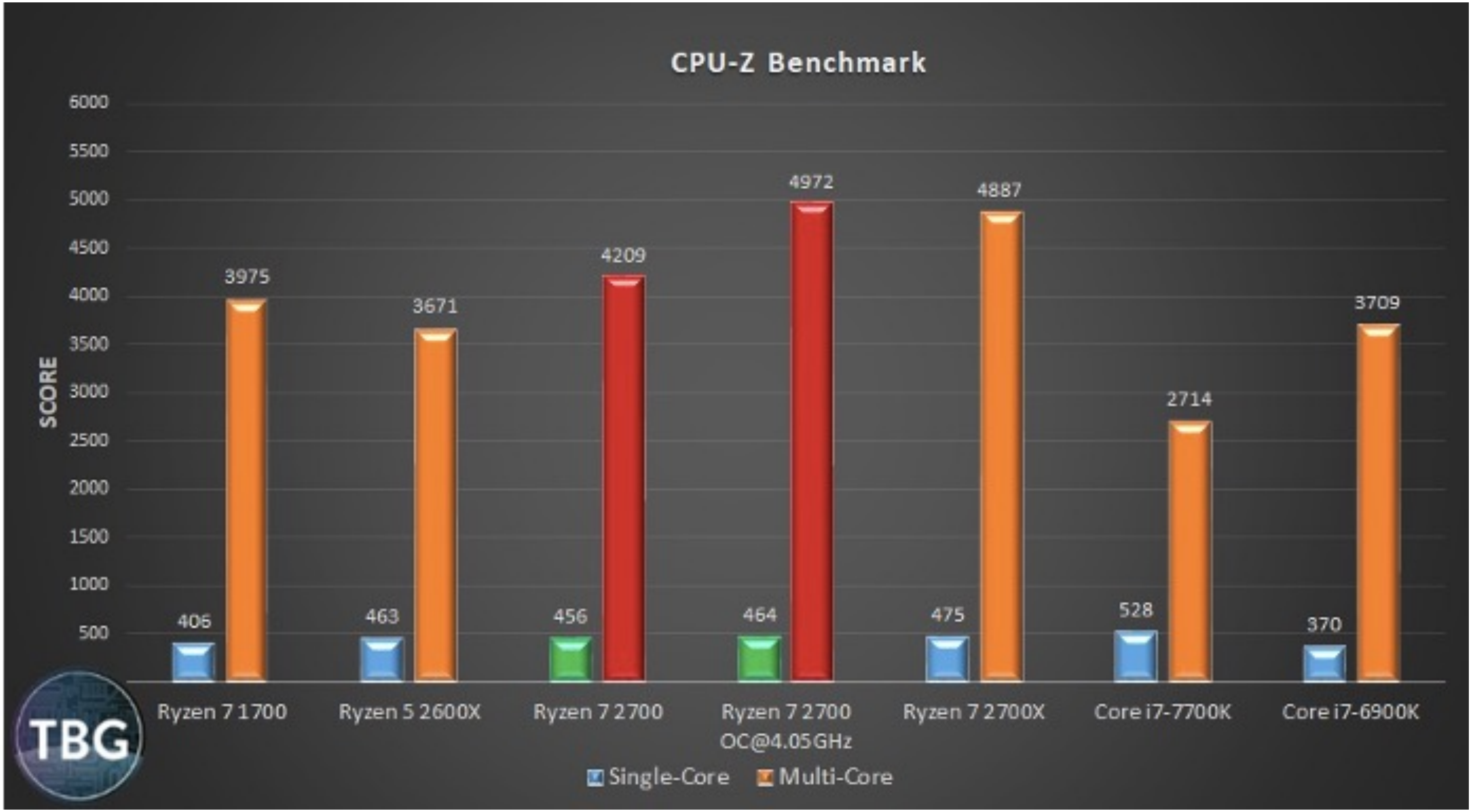


Misc Benchmarks

CPU Multi-Threaded Benchmark Hierarchy Post-Zen 3

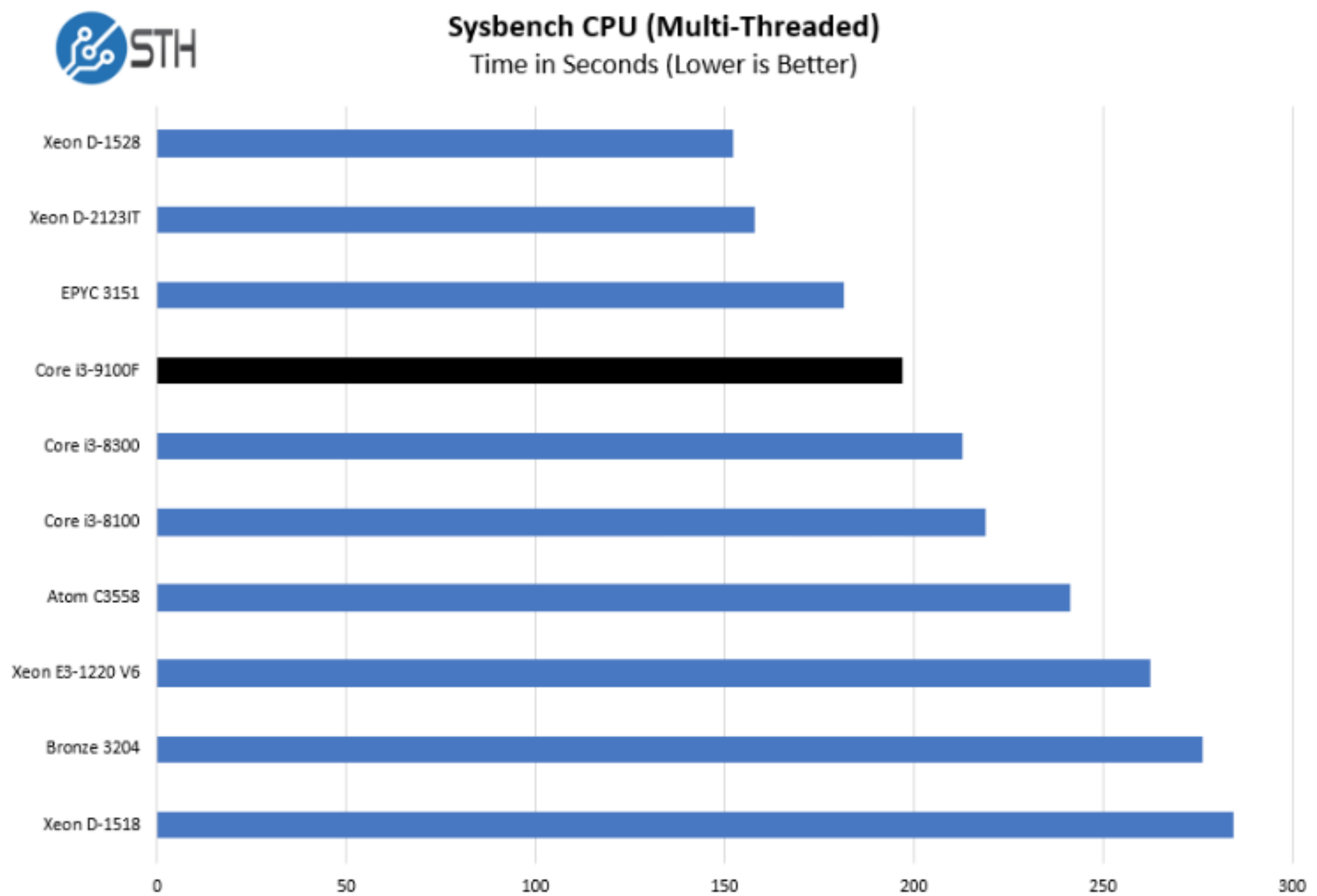
	Multi-Threaded App Score	CPU	Cores/Threads	Base/Boost	TDP
Threadripper 3960X	100%	Zen 2	24/48	3.8 / 4.5 GHz	280W
Ryzen 9 5950X	82.74%	Zen 3	16/32	3.4 / 4.9 GHz	105W
AMD Ryzen 9 3950X	73.07%	Zen 2	16/32	3.5 / 4.7 GHz	105W
Ryzen 9 5900X	70.87%	Zen 3	12/24	3.7 / 4.8 GHz	105W
Intel Core i9-10980XE	66.50%	Cascade Lake-X	18/36	3.0 / 4.8 GHz	165W
AMD Ryzen 9 3900X	59.75%	Zen 2	12/24	3.8 / 4.6 GHz	105W
AMD Ryzen 9 3900XT	59.69%	Zen 2	12/24	3.8 / 4.7 GHz	105W
Intel Core i9-10900K	54.16%	Comet Lake	10/20	3.7 / 5.3 GHz	125W

Misc Benchmarks



Misc Benchmarks

MT



Section

SMT **ON** vs **OFF**

MT Benchmarks

Investigating Performance of Multi-Threading on Zen 3 and AMD Ryzen 5000

by Dr. Ian Cutress on December 3, 2020 10:00 AM EST

https://www.anandtech.com/show/16261/investigating-performance-of-multithreading-on-zen-3-and-amd-ryzen-5000/2

"AMD Ryzen 9 3900X, SMT on vs SMT off, vs Intel 9900K", TechPowerUp, https://www.techpowerup.com/review/amd-ryzen-9-3900x-smt-off-vs-intel-9900k/

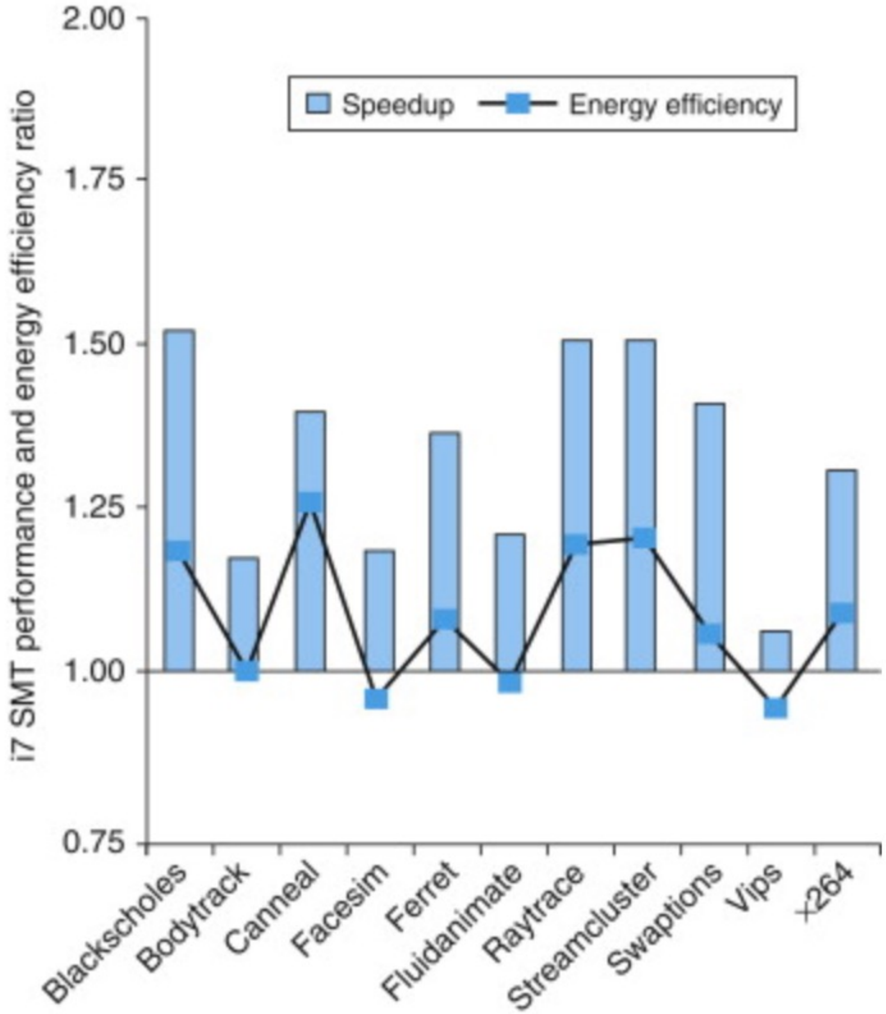
https://ece757.ece.wisc.edu/lect03-cores-multithread.pdf

SMT Performance

P&H zyBook

Figure 6.4.2: The speed-up from using multithreading on one core on an i7 processor (COD Figure 6.6).

Processor averages 1.31 for the PARSEC benchmarks (see COD Section 6.9 (Communicating to the outside world: Cluster networking)) and the energy efficiency improvement is 1.0



SMT in AMD Ryzen (Zen 3)

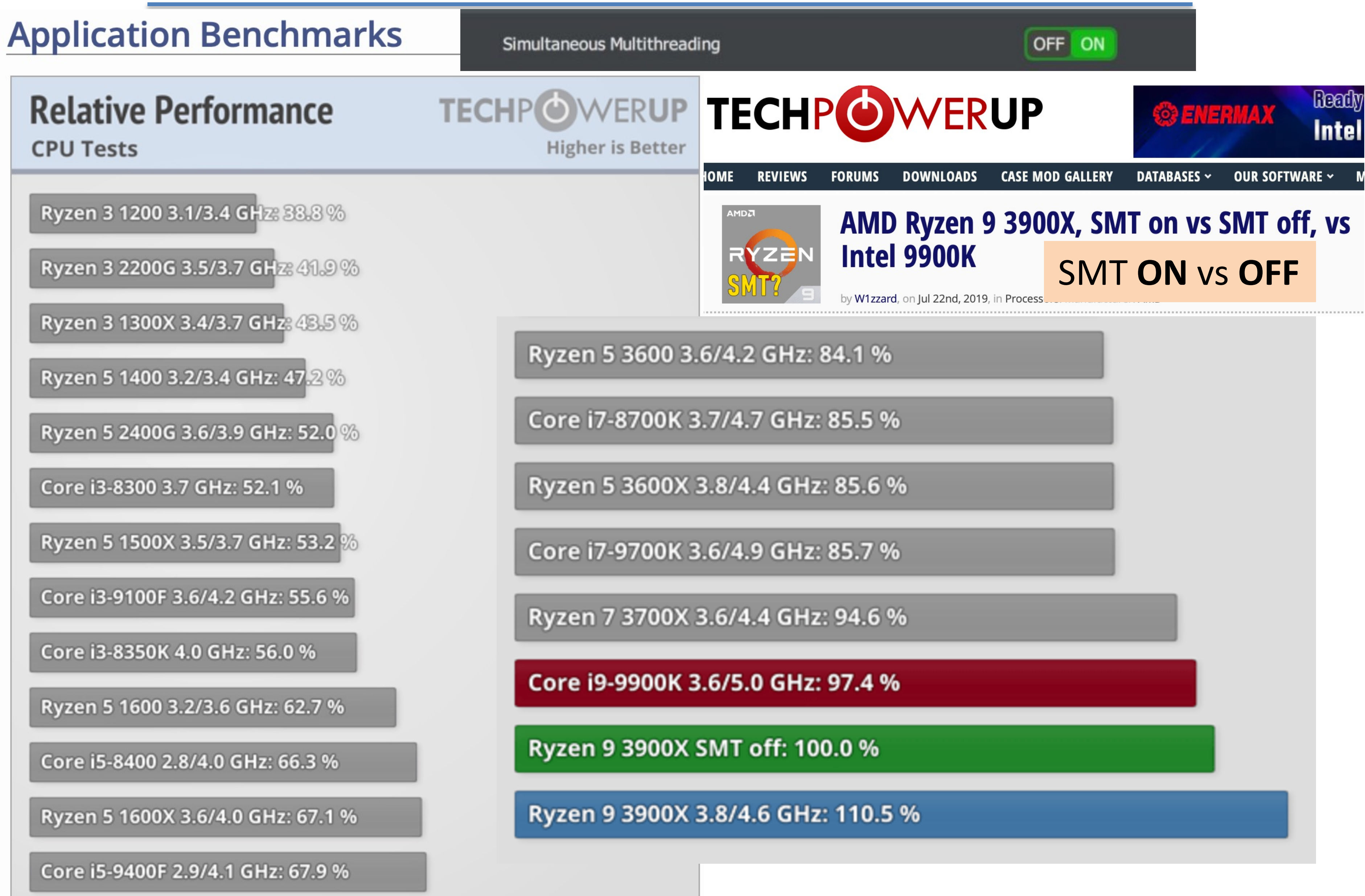
Simultaneous Multithreading OFF ON

Multi-Threaded Tests
AMD Ryzen 9 5950X

SMT ON vs OFF

AnandTech	SMT Off Baseline	SMT On
Agisoft Photoscan	100%	98.2%
3D Particle Movement	100%	165.7%
3DPM with AVX2	100%	177.5%
y-Cruncher	100%	94.5%
NAMD AVX2	100%	106.6%
AIBench	100%	88.2%
Blender	100%	125.1%
Corona	100%	145.5%
POV-Ray	100%	115.4%
V-Ray	100%	126.0%
CineBench R20	100%	118.6%
HandBrake 4K HEVC	100%	107.9%
7-Zip Combined	100%	133.9%
AES Crypto	100%	104.9%

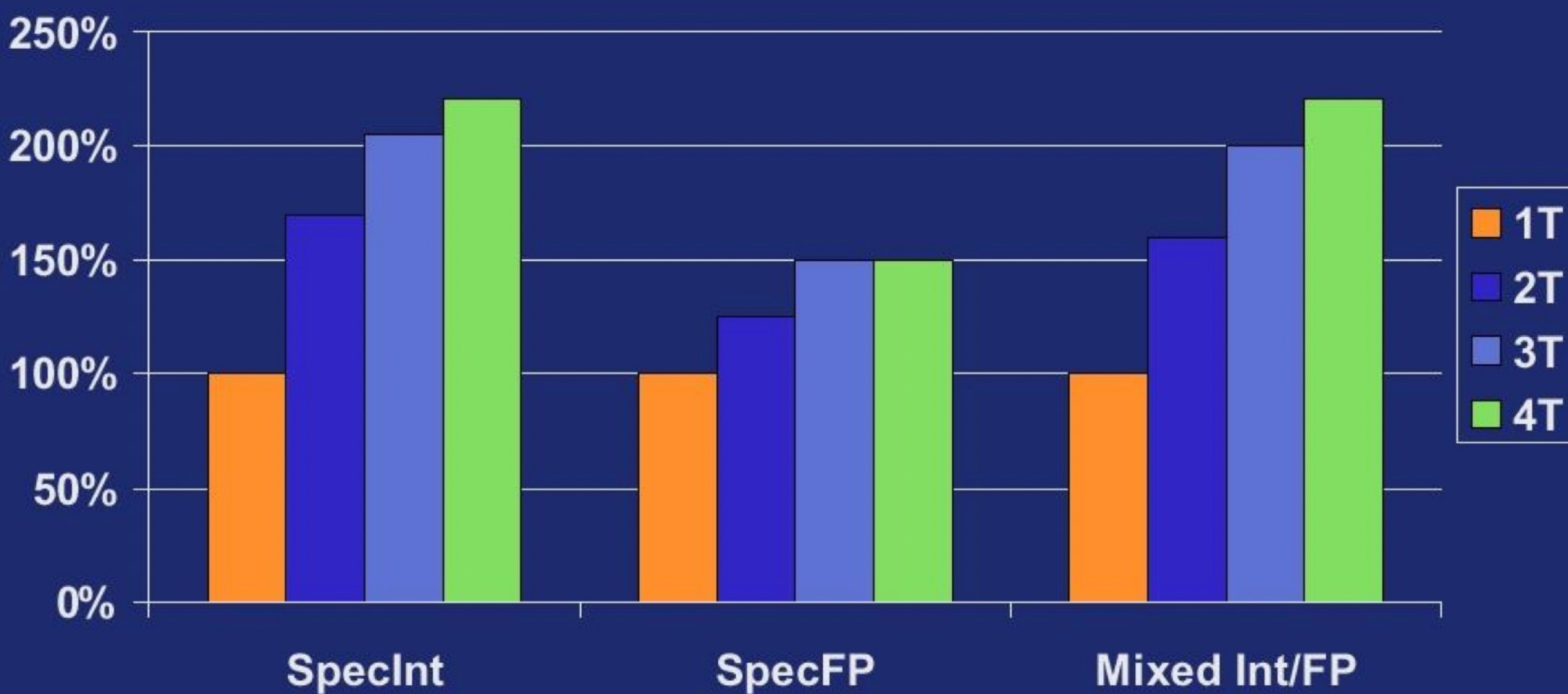
SMT Benchmarks



U Wisc Slides: Benchmarks

Mikko Lipasti-University of Wisconsin

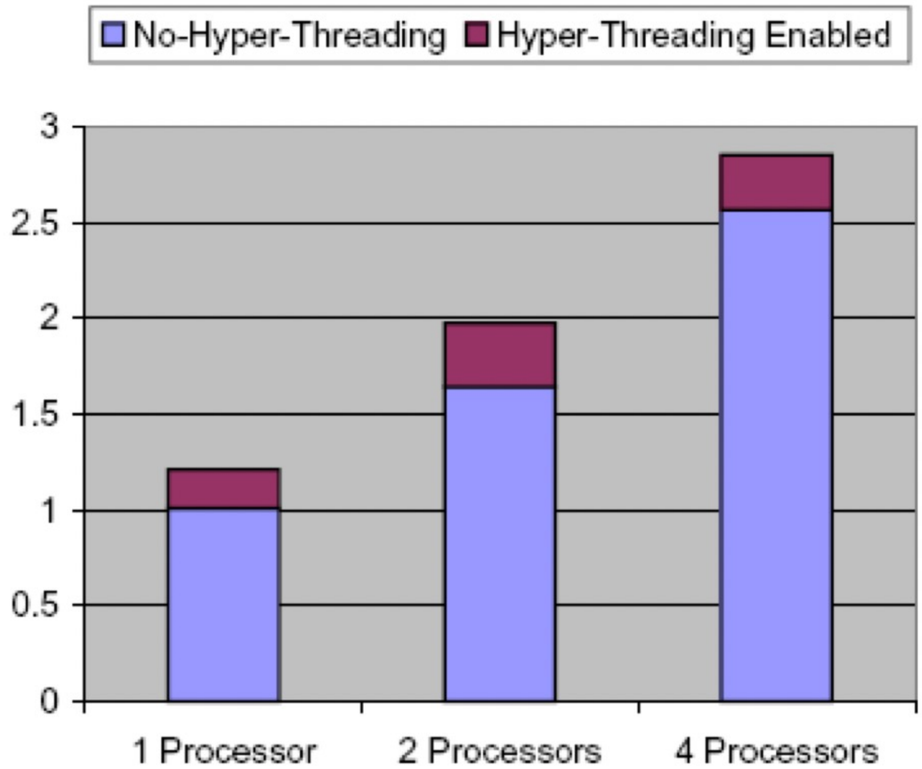
Multiprogrammed workload



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

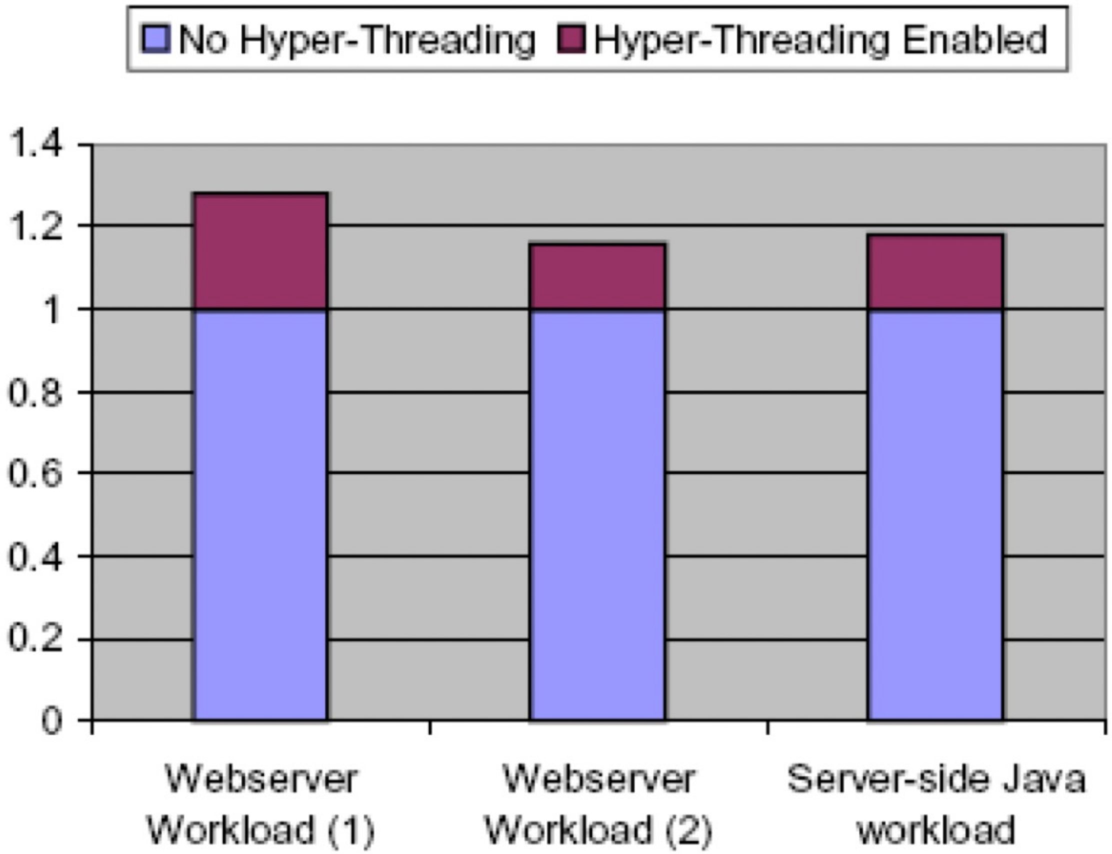
U Wisc Slides: Benchmarks

- OLTP workload
  - 21% gain in single and dual systems
  - Likely external bottleneck in 4 processor systems
    - Most likely front-side bus (FSB), i.e. memory bandwidth



U Wisc Slides: Benchmarks

- Web server apps



Section

Benchmark Cheating

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Cheat Sheet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

Androidpolice

NEWS · APP & GAMES · REVIEWS & GUIDES · DEVICES

HOME › NEWS

MediaTek has been caught cheating on benchmarks, and it's not pretty

If they could put that same effort into making better chips, we would all benefit

BY RYNE HAGER
PUBLISHED APR 08, 2020

26

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Cheat Sheet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

Quora 99 641 Search Quora

MediaTek has been caught cheating on benchmarks, and it's not pretty

If they could put that same effort into making better chips, we woul...

https://www.androidpolice.com/2020/04/08/mediatek-has-been-...

Phones we caught cheating benchmarks in 2018

Here's how companies cheat on benchmarks and how we caught...

https://www.androidauthority.com/the-companies-we-busted-ch...

Samsung owes Galaxy S4 owners $10 for cheating on benchmarks

Back in 2013, when the Galaxy S4 was the flagship of Samsung's...

https://www.androidpolice.com/2019/10/02/samsung-galaxy-s4-...

Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated

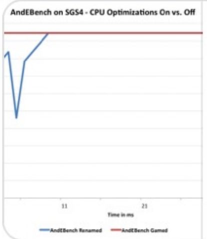
The OnePlus 5 is taking part in benchmark cheating again in an...

https://www.xda-developers.com/oneplus-5-benchmark-cheatin...

Cheat Sheet

Quora

Search Quora



They're (Almost) All Dirty: The State of Cheating in Android Benchmarks

Thanks to AndreiF7's excellent work on discovering it, we kicked off...

https://www.anandtech.com/show/7384/state-of-cheating-in-an...

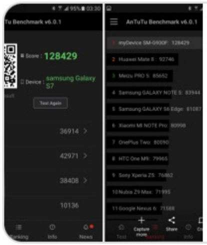
Huawei & Honor's Recent Benchmarking Behaviour: A Cheating Headache

https://www.anandtech.com/show/13318/huawei-benchmark-che...

Huawei's recent cheating wont help it win over Americans

Huawei's been caught faking phone benchmarks. The Chinese...

https://mashable.com/article/huawei-cheating-phone-benchmar...



Why and how do OEMs cheat on benchmarking? - Gary explains

Benchmark cheating is back in the news, this time the culprits are...

https://www.androidauthority.com/oems-cheat-on-benchmarkin...

Almost all Android devices cheat at benchmarks, report says

You upvoted this

Cheat Sheet

Quora   99   641   Search Quora

Almost all Android devices cheat at benchmarks, report says

The only Android manufacturers who aren't cheating at benchmarks...

https://www.cnet.com/news/almost-all-android-devices-cheat-at...

Android manufacturers just can’t stop cheating on benchmark tests

Even though they keep getting caught when they cheat on...

https://bgr.com/2018/09/07/huawei-p20-pro-benchmark-cheatin...

Popular Android manufacturer OnePlus caught cheating in benchmark tests

A few years ago, many high-profile Android device makers were...

https://bgr.com/2017/02/01/oneplus-3t-benchmark-cheating/

Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated

The OnePlus 5 is taking part in benchmark cheating again in an...

https://www.xda-developers.com/oneplus-5-benchmark-cheatin...

Section

Geekbench

Geekbench

MADE BY PRIMATE LABS

HOME BROWSER STORE BLOG SUPPORT

Geekbench

Geekbench is a cross-platform utility for benchmarking the central processing unit of computers.

Wikipedia Official site

Developers: Primate Labs Inc. · Primate Labs Inc

Written in: C++, C, Objective-C, Python, Ruby

Available in: English

Introducing Geekbench 5

Geekbench 5 is a cross-platform benchmark that measures your system's performance with the press of a button. How will your mobile device or desktop computer perform when push comes to crunch? How will it compare to the newest devices on the market? Find out today with Geekbench 5.

Download Buy Now

Geekbench 5 - Cross-Platform Benchmark

https://www.geekbench.com ▾

Test your system's potential for gaming, image processing, or video editing with the **Compute Benchmark**. Test your GPU's power with support for the OpenCL, CUDA, and Metal APIs. New to **Geekbench** 5 is support for Vulkan, the next-generation cross-platform graphics and compute API.

Geekbench

Geekbench Score

1179	8046
Single-Core Score	Multi-Core Score

Geekbench 5.0.0 Pro for macOS x86 (64-bit)

System Information

System Information	
Operating System	macOS 10.14.6 (Build 18G84)
Model	iMac (27-inch Retina Early 2019)
Model ID	iMac19,1
Motherboard	Apple Inc. Mac-AA95B1DDAB278B95 iMac19,1

Geekbench

AMD Ryzen 7 5800X Benchmarks

CPU Benchmark Scores

Single-Core Score	Multi-Core Score
1659	10349

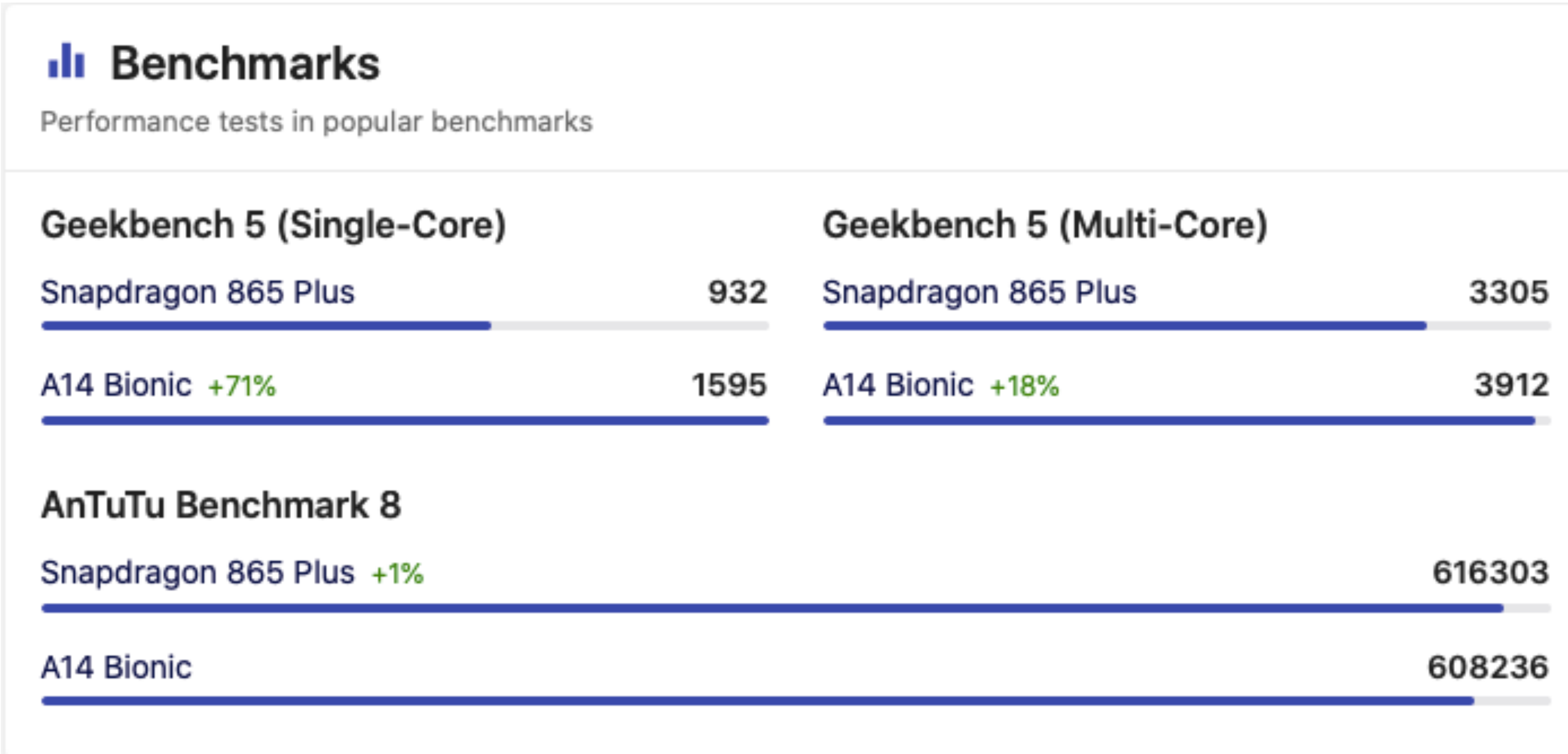
Intel Core i9-10900K Benchmarks

CPU Benchmark Scores

Single-Core Score	Multi-Core Score
1407	11012

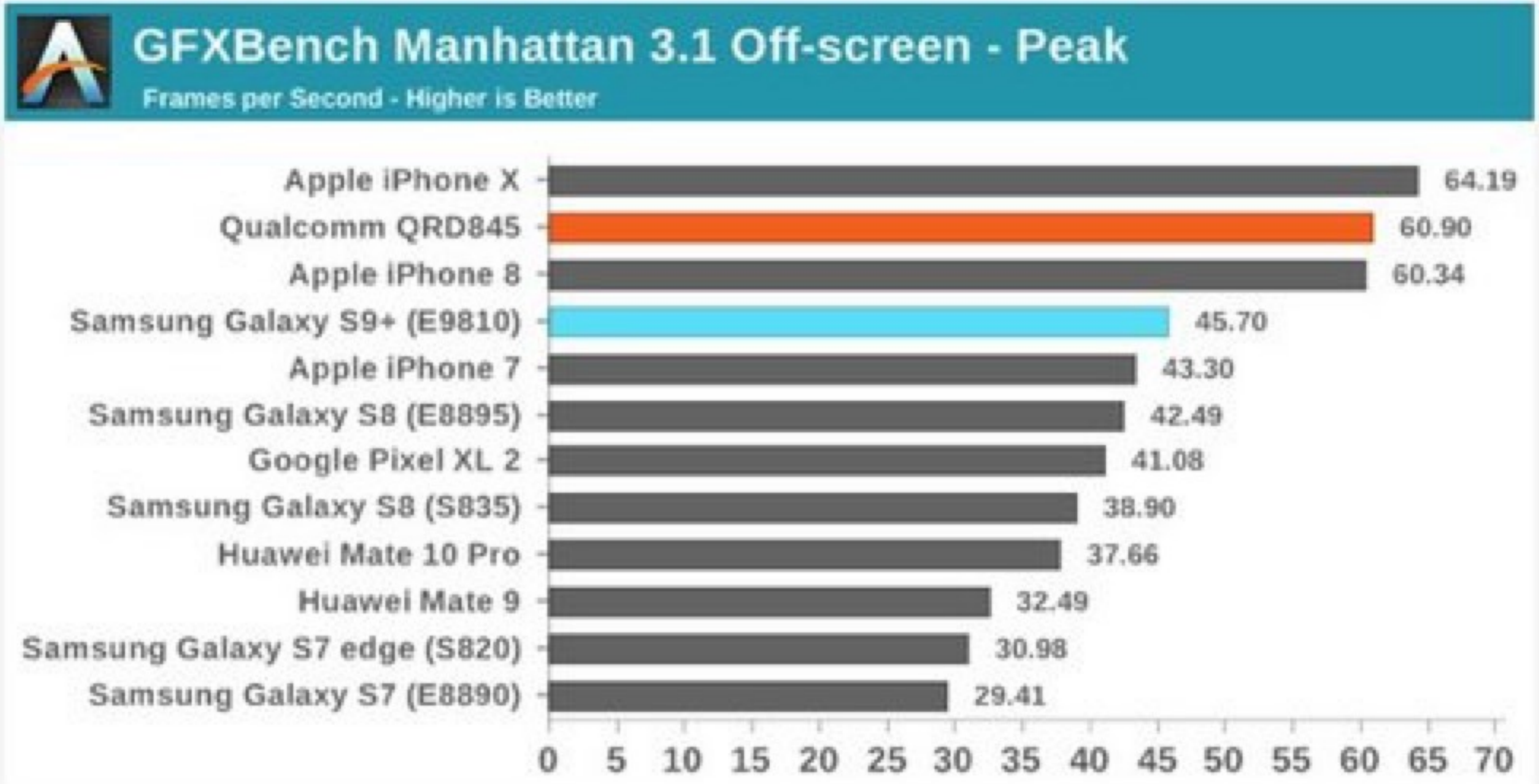
Single-core and Multi-Score Scores provided by Geekbench

Geekbench


Benchmarks
Performance tests in popular benchmarks
Geekbench 5 (Single-Core)
Snapdragon 865 Plus 932
A14 Bionic +71% 1595
Geekbench 5 (Multi-Core)
Snapdragon 865 Plus 3305
A14 Bionic +18% 3912
AnTuTu Benchmark 8
Snapdragon 865 Plus +1% 616303
A14 Bionic 608236

Amey Apte · Follow
Knows some things about them.

Benchmark



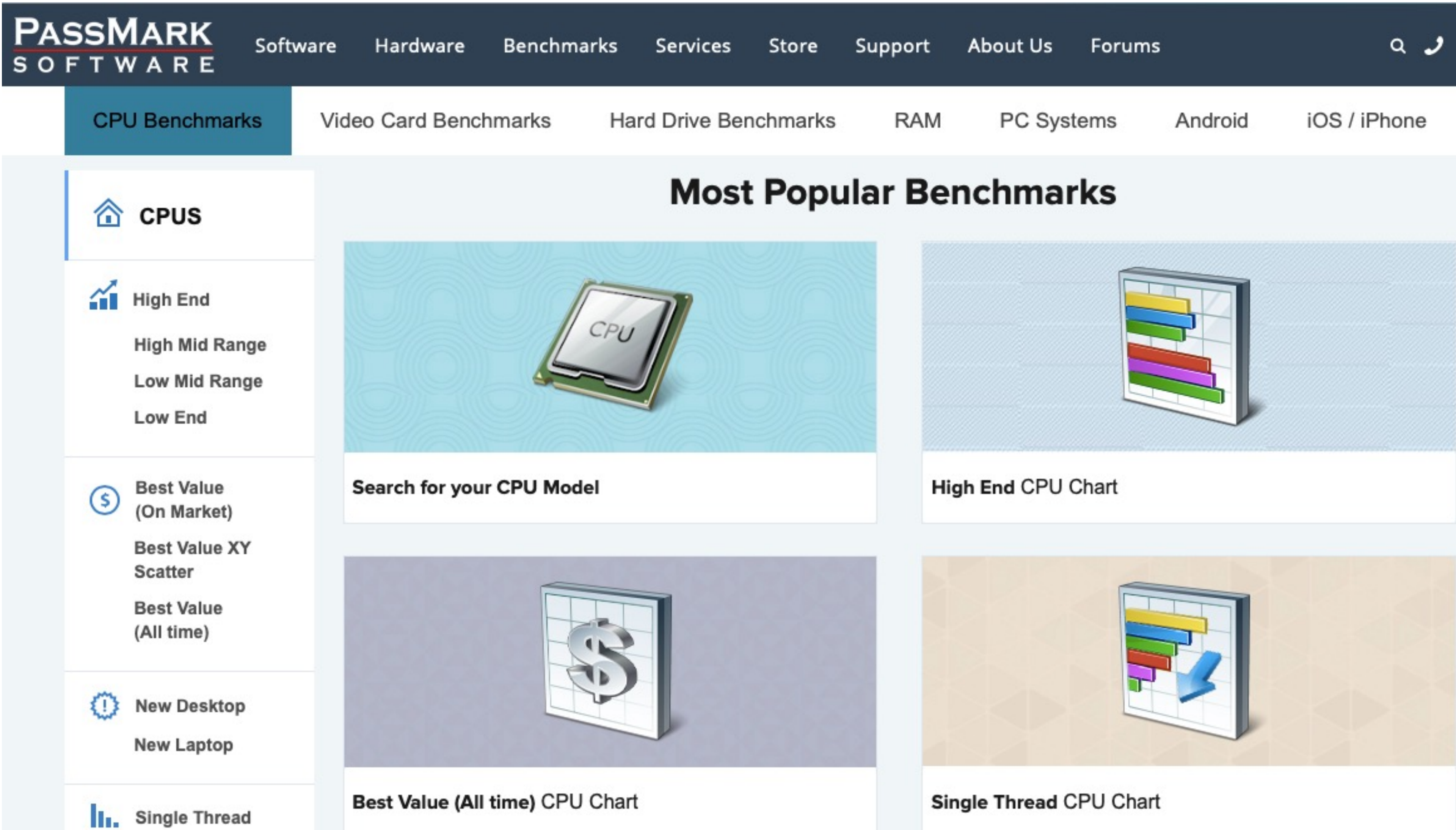
In Manhattan 3.1 the Exynos 9810 sees a mere 7% increase and lags far behind the new Snapdragon 845's Adreno 630.

Section

PassMark

https://www.cpubenchmark.net

PassMark

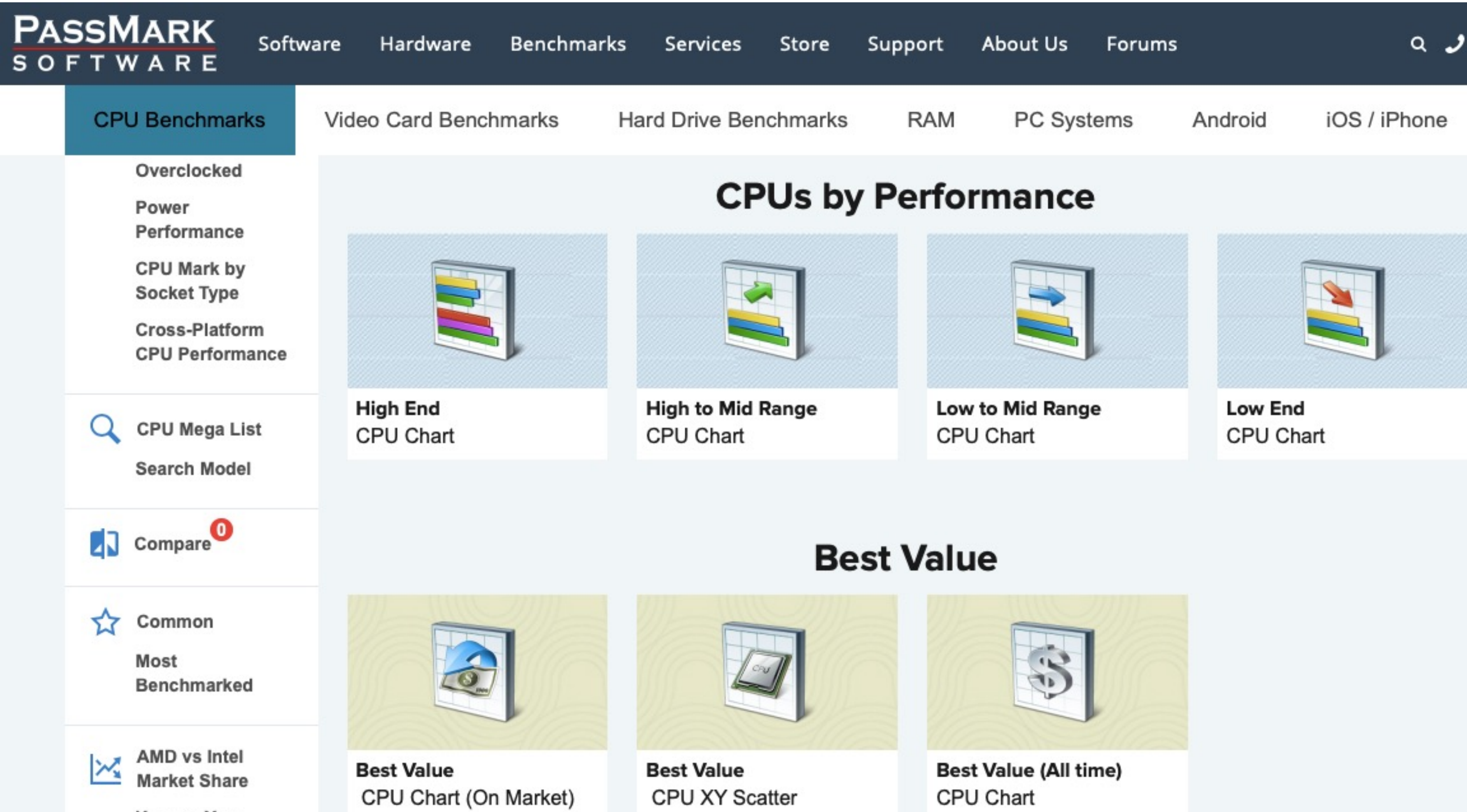

PassMark Software
Software
Hardware
Benchmarks
Services
Store
Support
About Us
Forums
CPU Benchmarks
Video Card Benchmarks
Hard Drive Benchmarks
RAM
PC Systems
Android
iOS / iPhone
CPUS
High End
High Mid Range
Low Mid Range
Low End
Best Value (On Market)
Best Value XY Scatter
Best Value (All time)
New Desktop
New Laptop
Single Thread
Most Popular Benchmarks
CPU
Search for your CPU Model
High End CPU Chart
Best Value (All time) CPU Chart
Single Thread CPU Chart

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

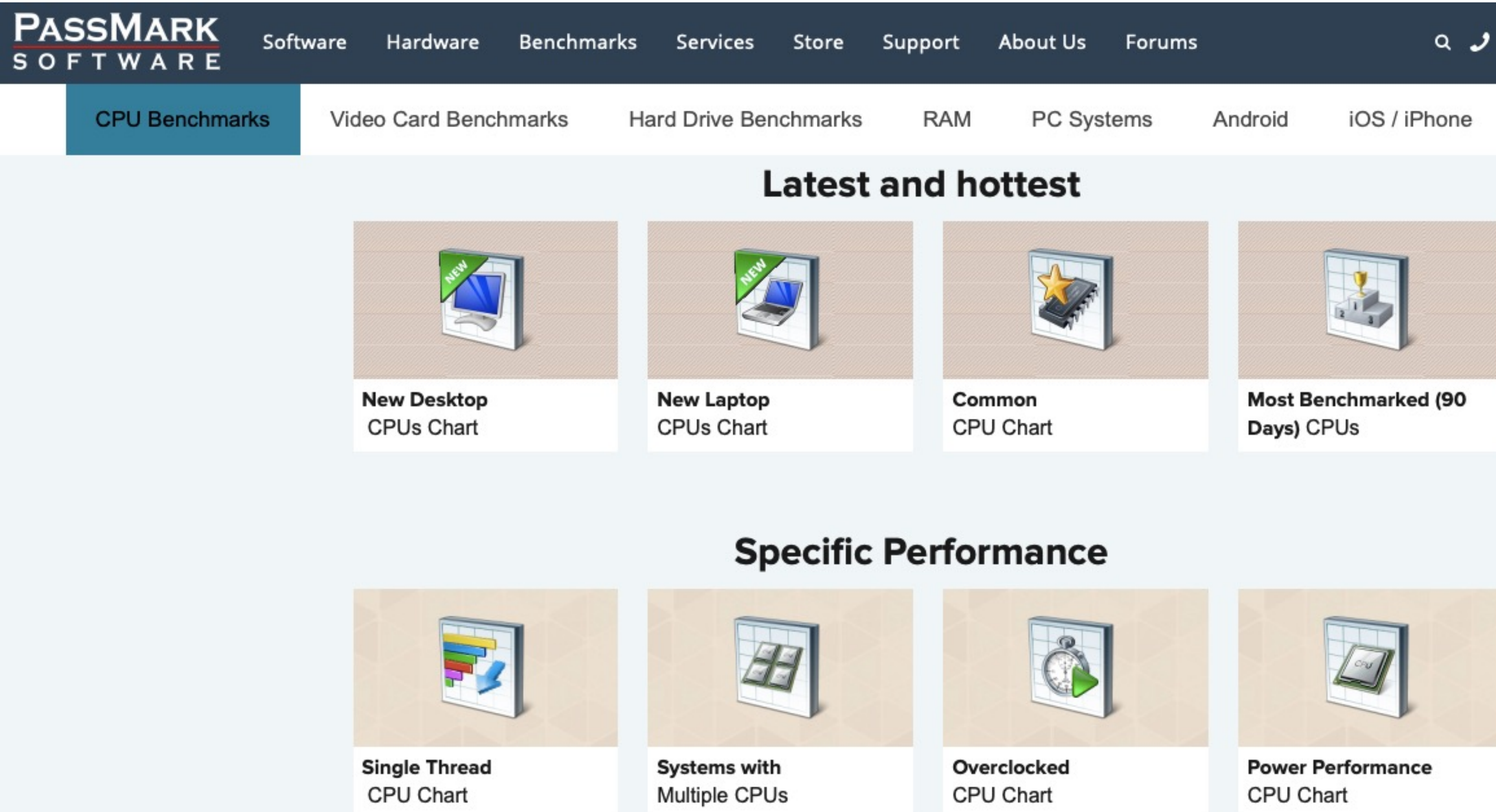
COMP222

PassMark

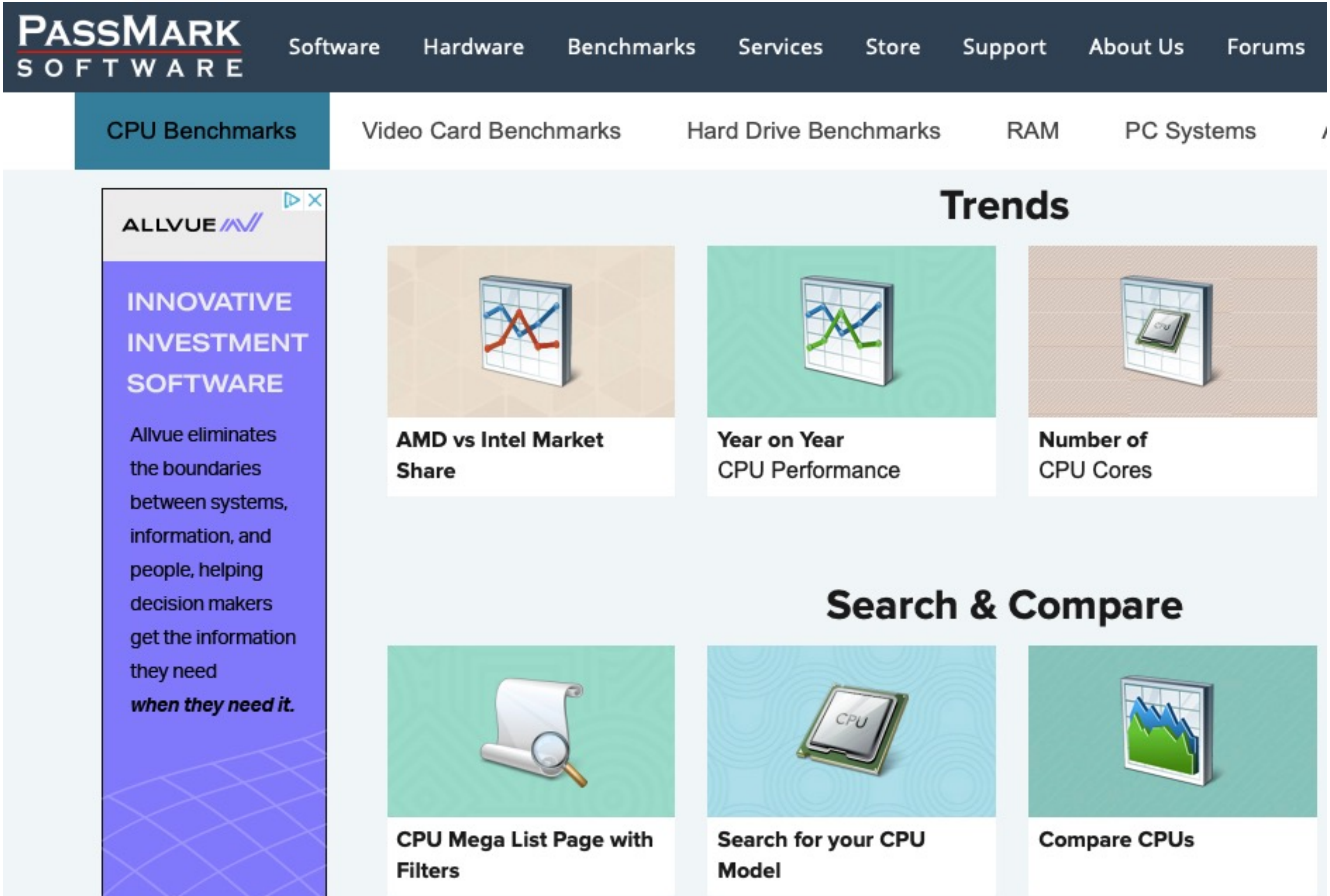
DR JEFF SOFTWARE
INDIE APP DEVELOPER



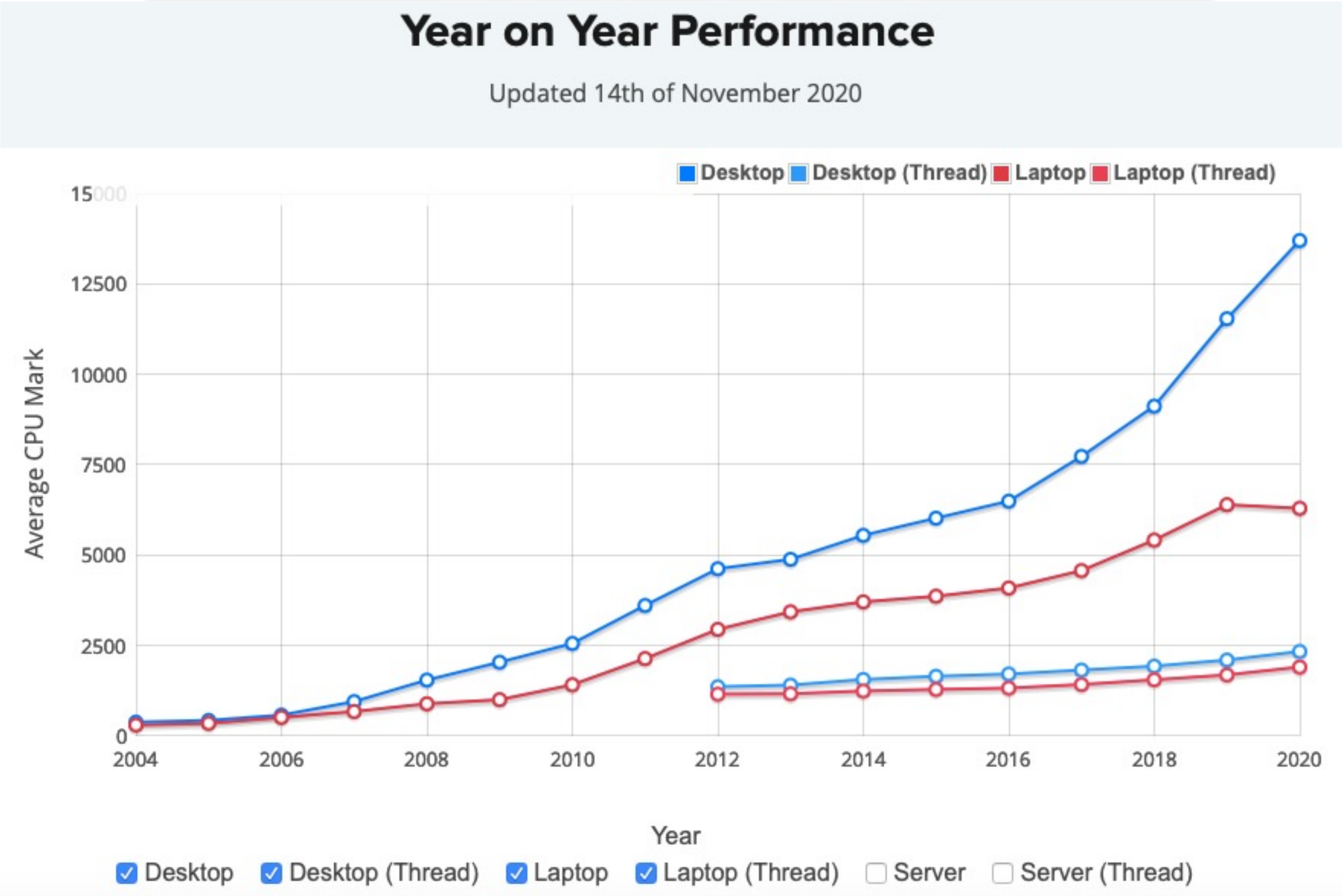
PassMark


PASSMARK SOFTWARE
Software
Hardware
Benchmarks
Services
Store
Support
About Us
Forums
CPU Benchmarks
Video Card Benchmarks
Hard Drive Benchmarks
RAM
PC Systems
Android
iOS / iPhone
Latest and hottest
NEW
New Desktop CPUs Chart
NEW
New Laptop CPUs Chart
Common CPU Chart
Most Benchmarked (90 Days) CPUs
Specific Performance
Single Thread CPU Chart
Systems with Multiple CPUs
Overclocked CPU Chart
CPU
Power Performance CPU Chart

PassMark


PASSMARK SOFTWARE
Software
Hardware
Benchmarks
Services
Store
Support
About Us
Forums
CPU Benchmarks
Video Card Benchmarks
Hard Drive Benchmarks
RAM
PC Systems
ALLVUE
INNOVATIVE INVESTMENT SOFTWARE
Allvue eliminates the boundaries between systems, information, and people, helping decision makers get the information they need when they need it.
Trends
AMD vs Intel Market Share
Year on Year CPU Performance
Number of CPU Cores
Search & Compare
CPU Mega List Page with Filters
Search for your CPU Model
Compare CPUs

PassMark


Year on Year Performance
Updated 14th of November 2020
Desktop
Desktop (Thread)
Laptop
Laptop (Thread)
Average CPU Mark
15000
12500
10000
7500
5000
2500
0
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year
Desktop
Desktop (Thread)
Laptop
Laptop (Thread)
Server
Server (Thread)

PassMark

PassMark - CPU Mark

High End CPUs

Updated 24th of November 2020

CPU	CPU Mark	Price (USD)
AMD Ryzen Threadripper PRO 3995WX	88,609	NA
AMD Ryzen Threadripper 3990X	80,783	$3,799.99
AMD EPYC 7702	71,859	$5,350.00
AMD EPYC 7702P	68,213	$4,430.00
AMD EPYC 7742	67,185	$6,850.00
AMD Ryzen Threadripper PRO 3975WX	65,674	NA
AMD Ryzen Threadripper 3970X	64,238	$2,326.87
AMD Ryzen Threadripper 3960X	55,391	$1,349.99
AMD EPYC 7502	53,591	$2,600.00
AMD Ryzen 9 5950X	46,724	$1,499.00
AMD EPYC 7502P	46,180	$2,524.77
AMD EPYC 7452	45,056	$2,288.99
AMD Ryzen Threadripper PRO 3955WX	42,095	NA
AMD Ryzen 9 5900X	39,667	$549.99
Intel Xeon W-3275M @ 2.50GHz	39,478	$7,453.00*
AMD Ryzen 9 3950X	39,239	$709.99
AMD EPYC 7402P	39,118	$1,435.99

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

PassMark

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

PassMark - CPU Mark

Multiple CPU Systems

Updated 8th of December 2020

CPU	CPU Mark	Price (USD)
[Dual CPU] AMD EPYC 7552	91,019	$8,917.14
[Dual CPU] AMD EPYC 7702	87,034	$9,910.00
[Dual CPU] AMD EPYC 7742	75,756	$13,580.00
[Dual CPU] AMD EPYC 7542	69,618	$8,190.00
[Dual CPU] Intel Xeon Gold 6258R @ 2.70GHz	64,315	$7,900.00*
[Dual CPU] AMD EPYC 7402	62,799	$3,969.98
[Dual CPU] Intel Xeon Platinum 8280 @ 2.70GHz	59,204	$19,898.00*
[Dual CPU] Intel Xeon Platinum 8171M @ 2.60GHz	57,780	NA
[Dual CPU] Intel Xeon Platinum 8260M @ 2.30GHz	55,297	$15,410.00*
[Dual CPU] AMD EPYC 7601	54,720	$11,076.00
[Dual CPU] Intel Xeon Gold 6240R @ 2.40GHz	54,024	$5,239.90*
[Dual CPU] Intel Xeon Gold 6254 @ 3.10GHz	50,901	$7,239.90*
[Dual CPU] Intel Xeon Platinum 8173M @ 2.00GHz	49,039	NA
[Quad CPU] Intel Xeon Platinum 8180M @ 2.50GHz	48,905	NA
[Dual CPU] Intel Xeon Platinum 8260 @ 2.40GHz	48,091	$9,220.00*

PassMark

PassMark - CPU Mark

Single Thread Performance

Updated 8th of December 2020

CPU	CPU Mark	Price (USD)
AMD Ryzen 9 5900X	3,521	$1,099.99
AMD Ryzen 9 5950X	3,521	$2,000.00
AMD Ryzen 7 5800X	3,512	$449.99
AMD Ryzen 5 5600X	3,391	$299.00*
Intel Core i9-10900K @ 3.70GHz	3,175	$579.99
Intel Core i9-10900KF @ 3.70GHz	3,150	$529.99
Intel Core i9-10900 @ 2.80GHz	3,111	$519.00
Intel Core i9-9990XE @ 4.00GHz	3,104	NA
Intel Core i9-10850K @ 3.60GHz	3,102	$439.99
Intel Core i9-10910 @ 3.60GHz	3,101	NA
Intel Core i9-10900F @ 2.80GHz	3,099	$419.99
Intel Core i7-10700KF @ 3.80GHz	3,090	$359.99
Intel Core i7-10700K @ 3.80GHz	3,088	$359.99
Intel Core i9-9900KS @ 4.00GHz	3,036	$1,965.00*
Intel Core i9-9900KF @ 3.60GHz	3,004	$329.99
Intel Core i9-9900K @ 3.60GHz	2,977	$362.95
Intel Core i5-10600KF @ 4.10GHz	2,970	$244.99
Intel Core i7-9700KF @ 3.60GHz	2,949	$299.00
Intel Core i5-10600K @ 4.10GHz	2,938	$267.66
Intel Core i7-10700 @ 2.90GHz	2,936	$349.99

PassMark

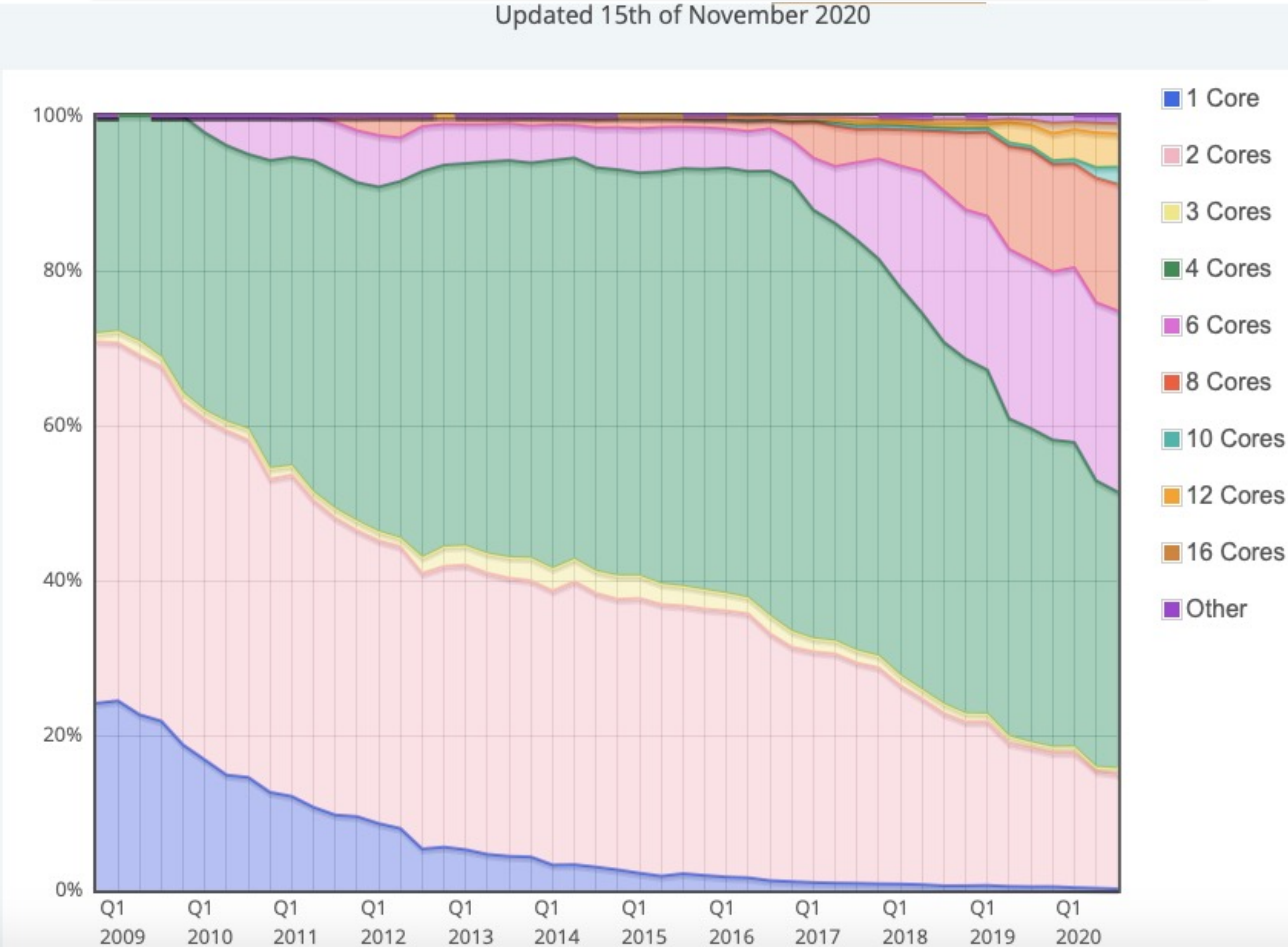
Number of CPU Cores

Updated 24th of November 2020

Number of CPU Cores	Percentage	Change
1 Core	0.18%	-0.02%
2 Cores	14.22%	-0.67%
3 Cores	0.60%	-0.17%
4 Cores	35.19%	-1.92%
6 Cores	23.81%	1.29%
8 Cores	16.72%	0.76%
10 Cores	2.10%	-0.18%
12 Cores	4.63%	0.55%
16 Cores	1.42%	0.18%
32 Cores	0.31%	0.06%
64 Cores	0.15%	0.04%
Other	0.67%	0.10%

- This chart only includes x86 processors and does not include other chip architectures.
- This chart only includes CPUs installed into PCs and does not include game consoles.

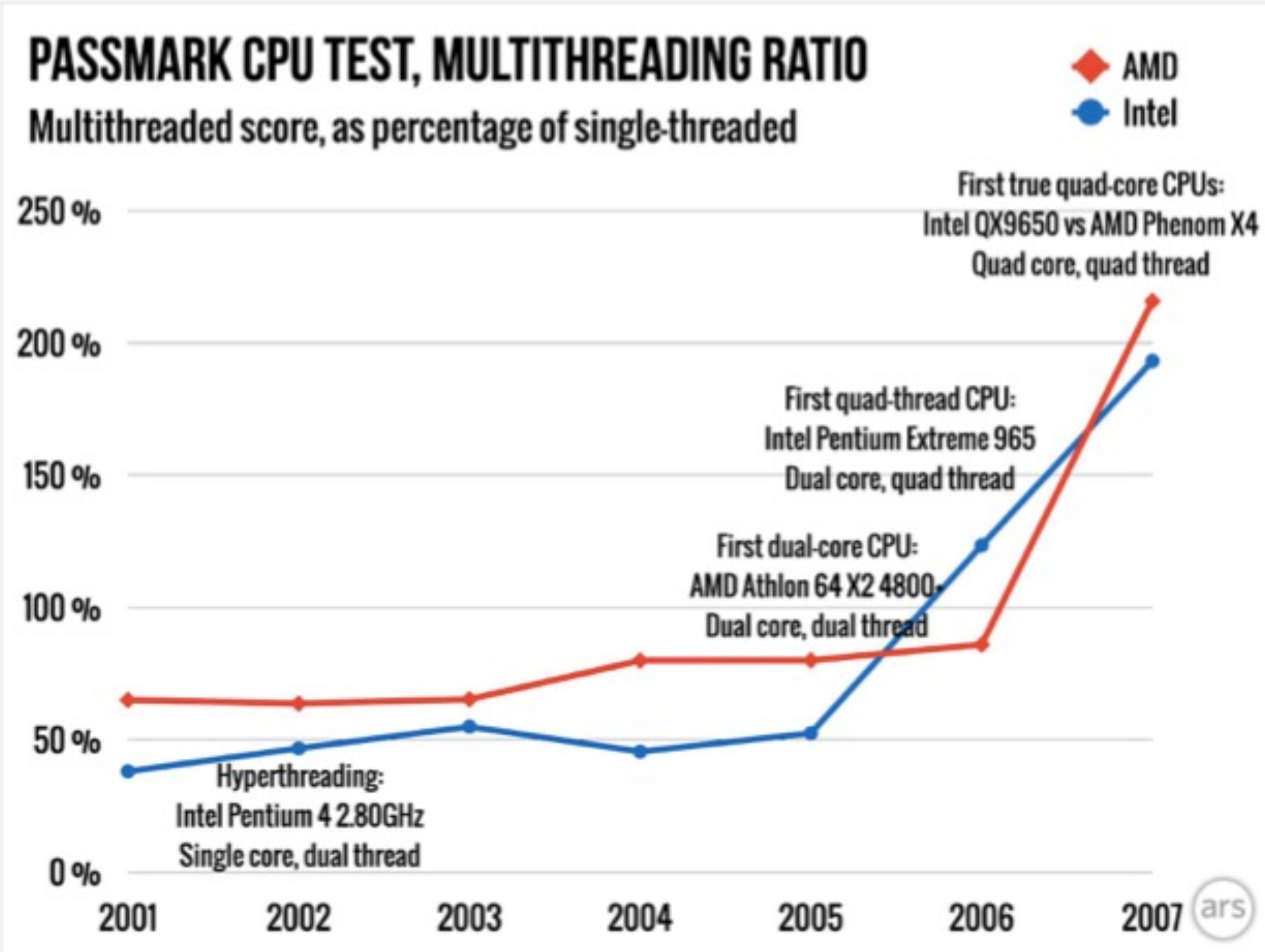
PassMark


Updated 15th of November 2020
100%
80%
60%
40%
20%
0%
Q1 2009
Q1 2010
Q1 2011
Q1 2012
Q1 2013
Q1 2014
Q1 2015
Q1 2016
Q1 2017
Q1 2018
Q1 2019
Q1 2020
1 Core
2 Cores
3 Cores
4 Cores
6 Cores
8 Cores
10 Cores
12 Cores
16 Cores
Other

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

PassMark

DR JEFF SOFTWARE
INDIE APP DEVELOPER



Section

Techspot

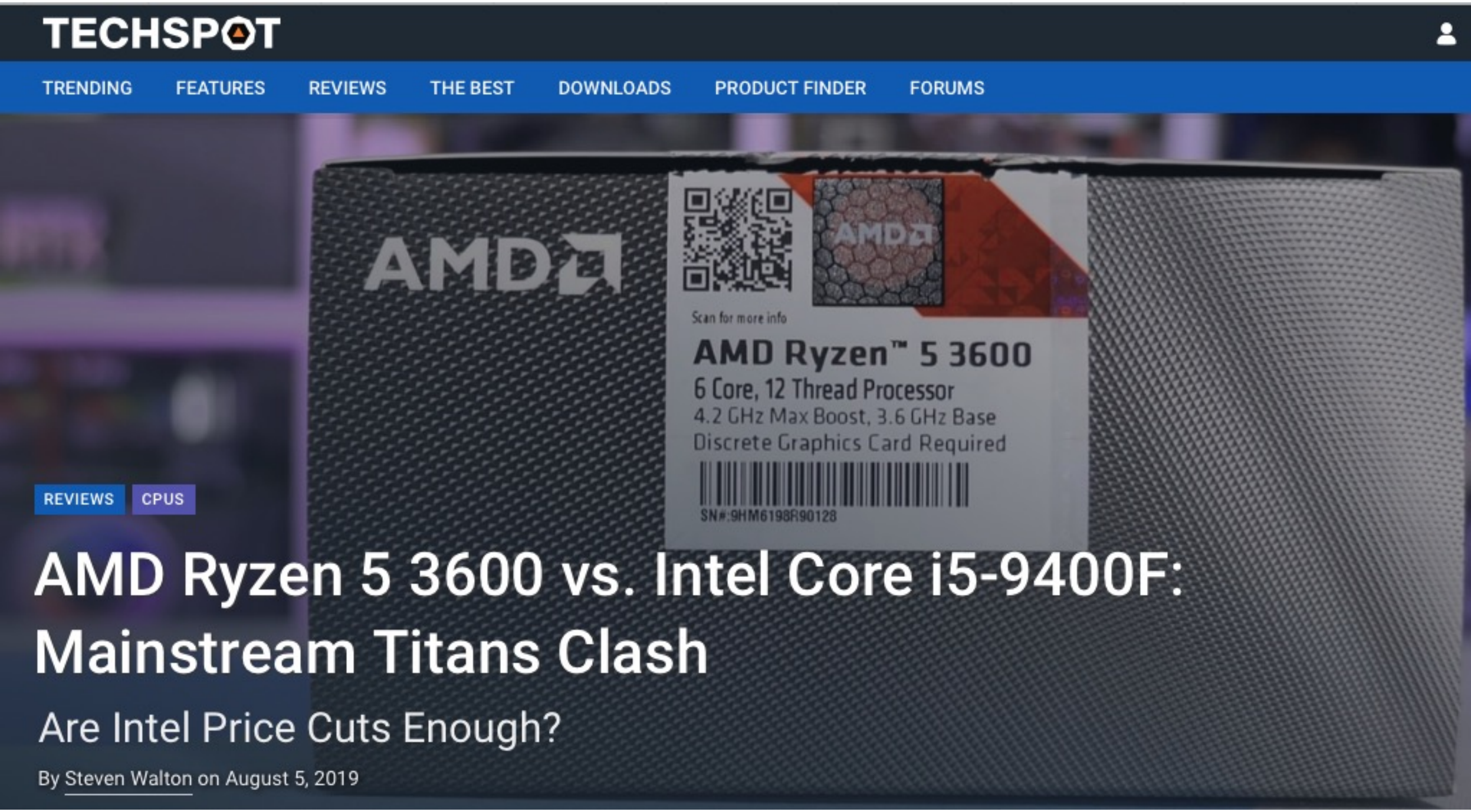
https://www.techspot.com/review/1885-ryzen-5-3600-vs-core-i5-9400f/

TECHSPOT

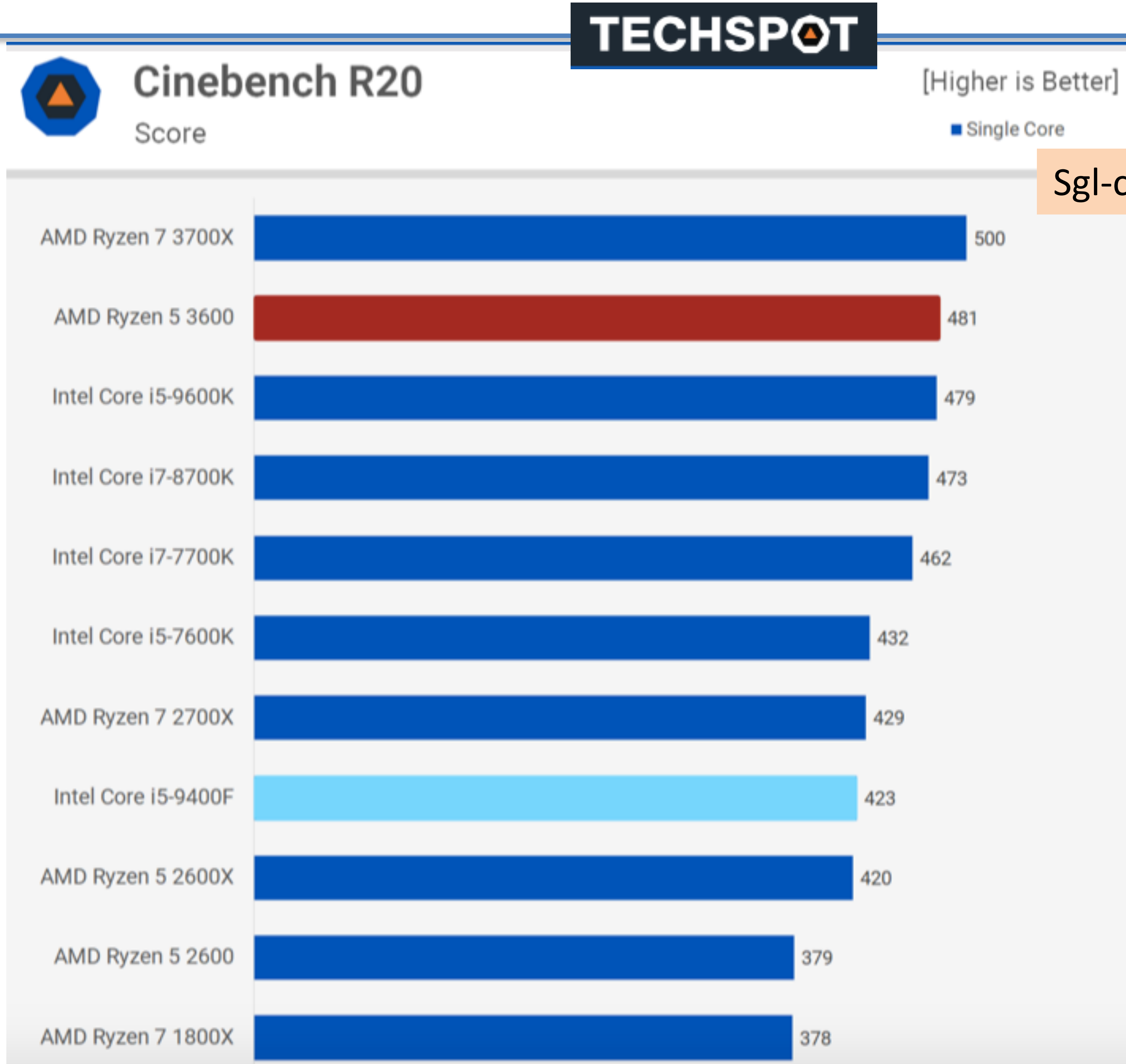
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

AMD Ryzen 5 vs Core i5

DR JEFF SOFTWARE
INDIE APP DEVELOPER

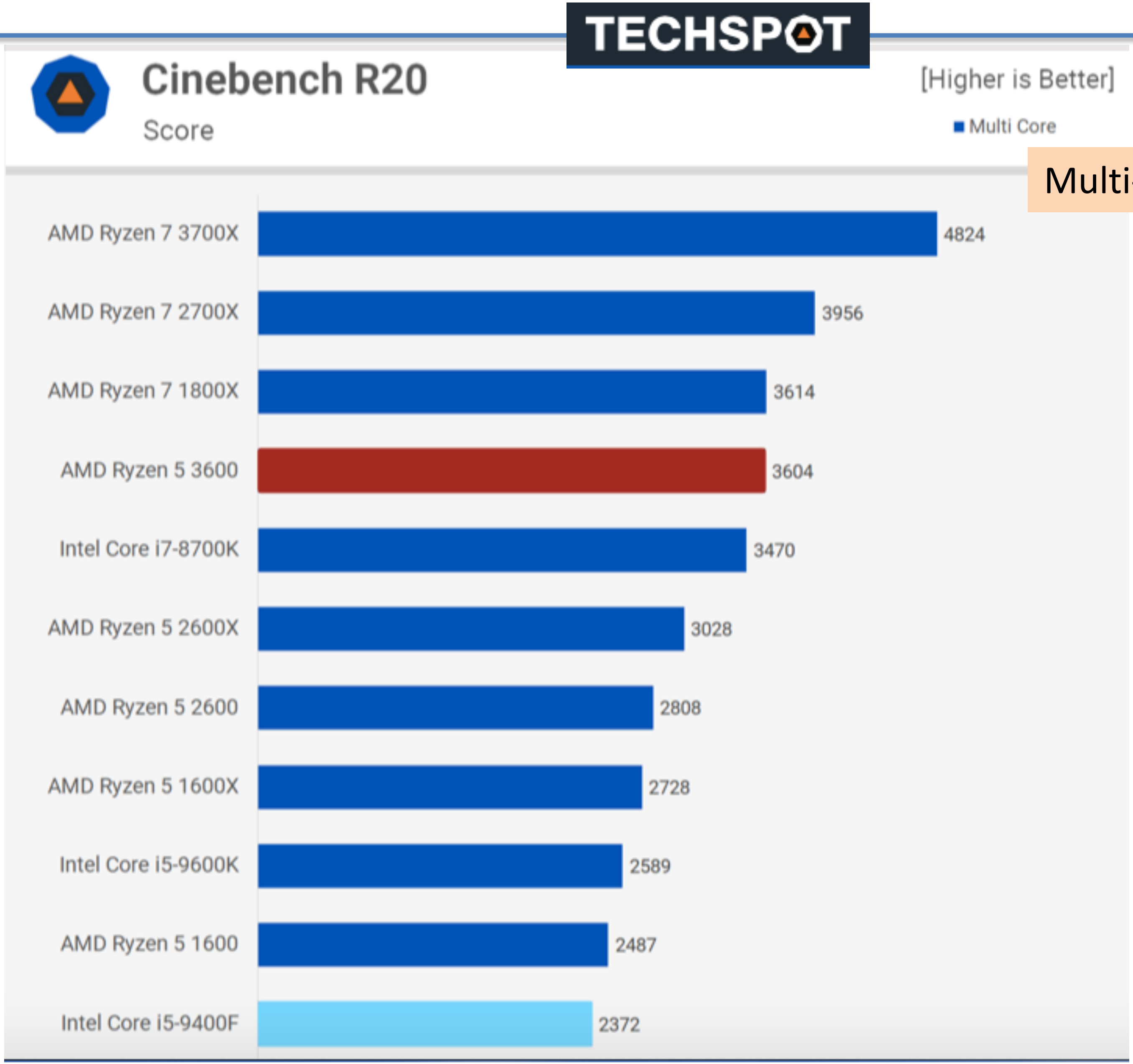


Techspot



Sgl-core

Techspot

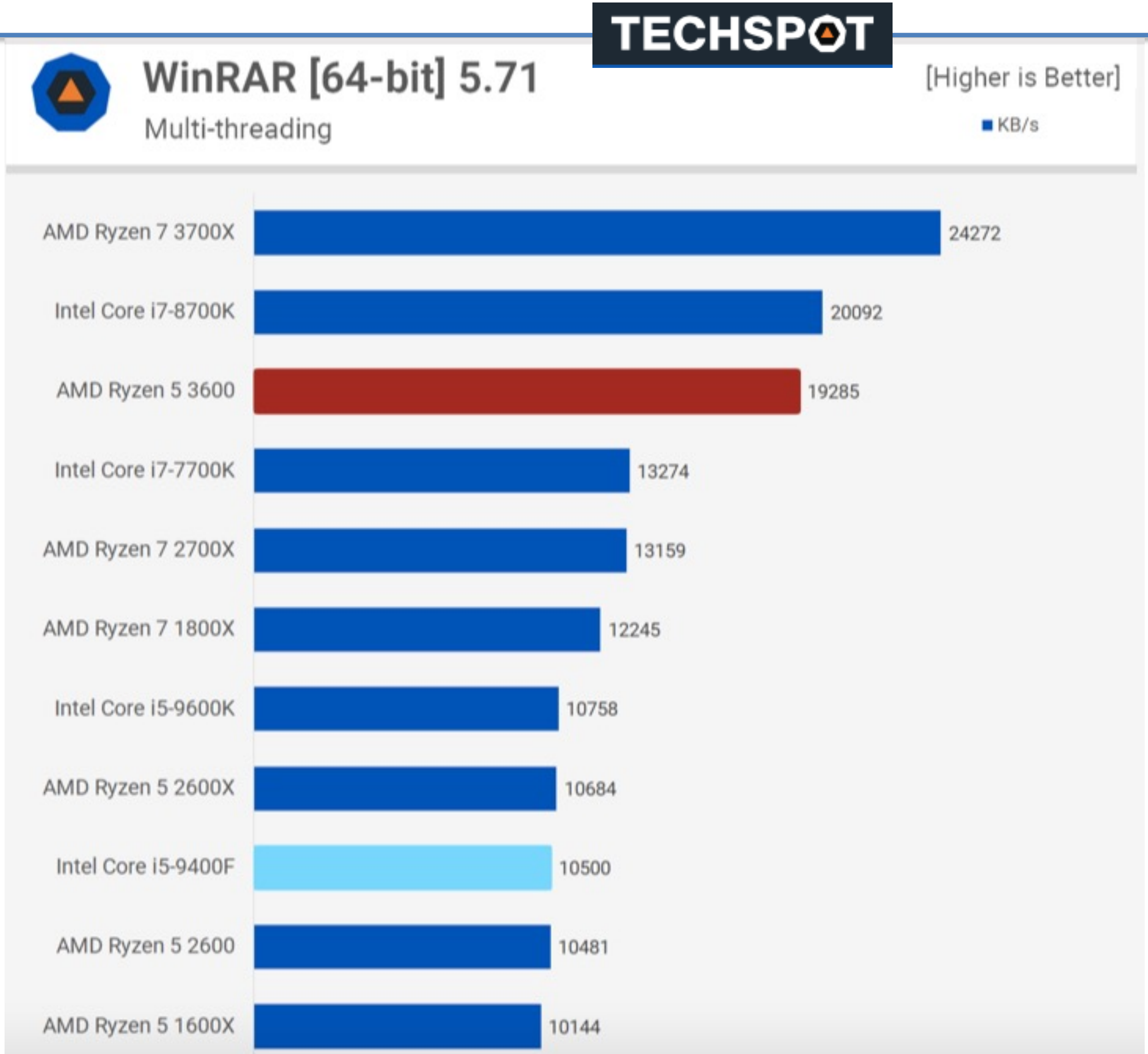


Multi-core

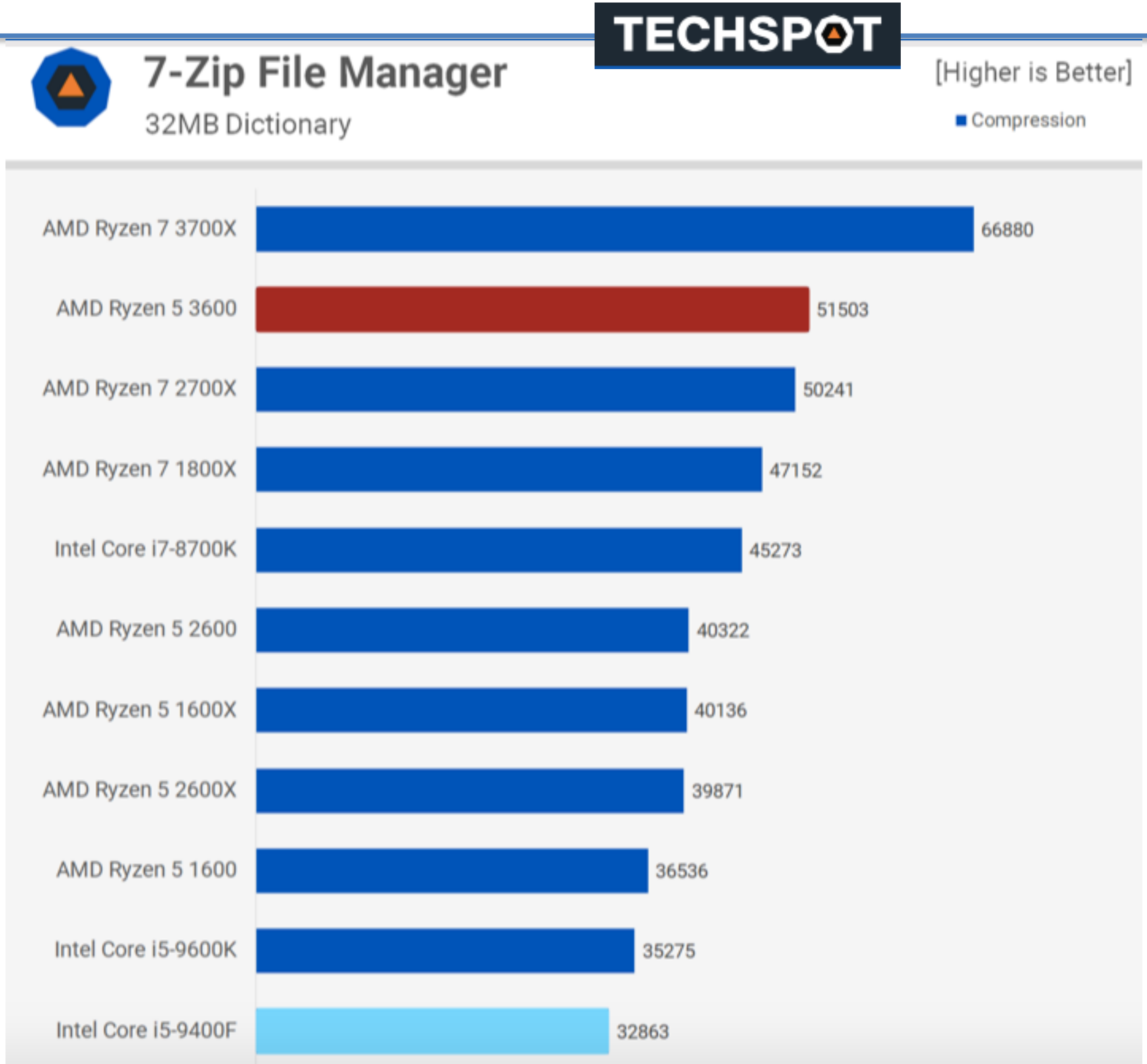
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Techspot

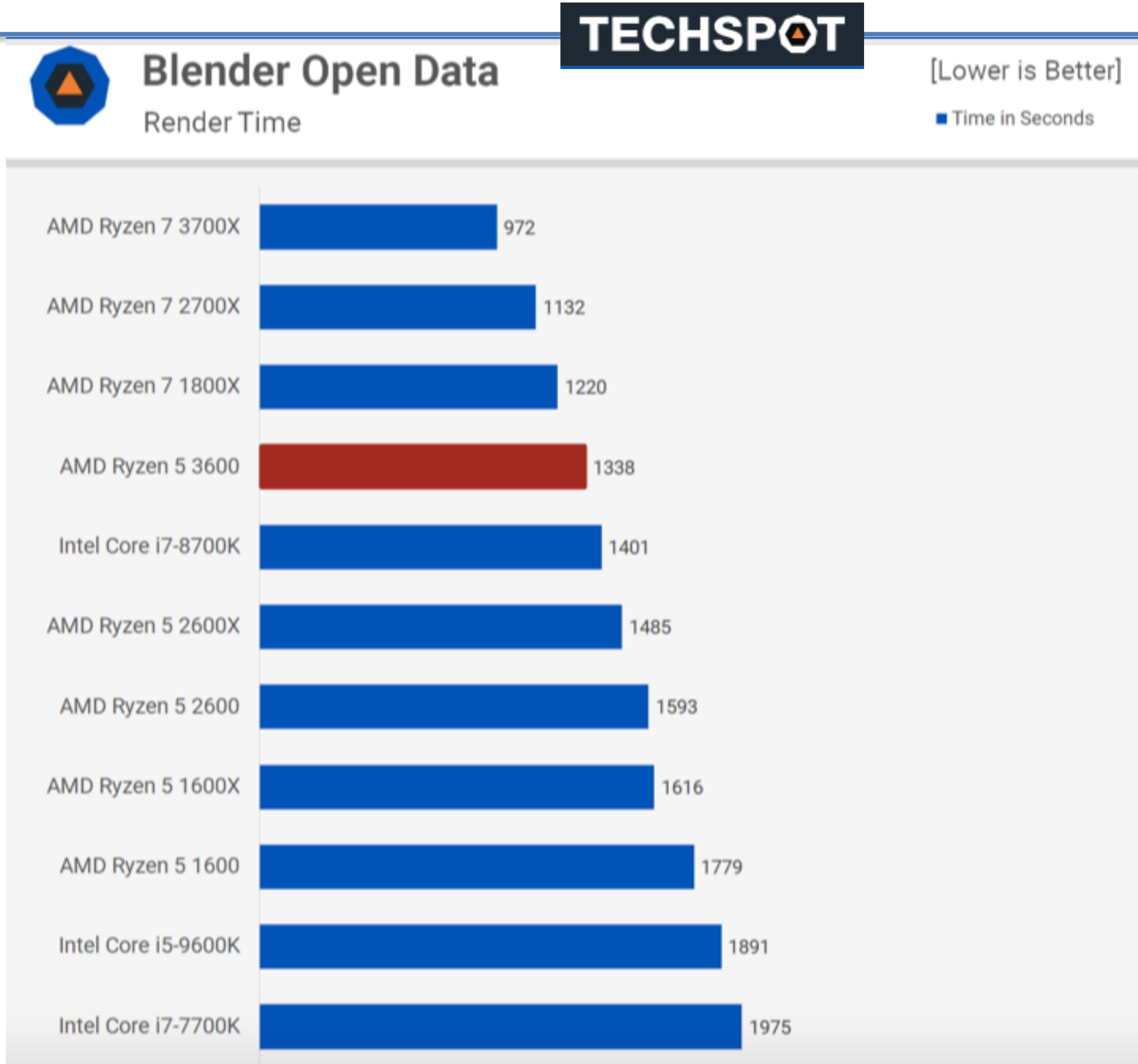
DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22



Techspot


TECHSPOT
7-Zip File Manager
32MB Dictionary
[Higher is Better]
Compression
AMD Ryzen 7 3700X 66880
AMD Ryzen 5 3600 51503
AMD Ryzen 7 2700X 50241
AMD Ryzen 7 1800X 47152
Intel Core i7-8700K 45273
AMD Ryzen 5 2600 40322
AMD Ryzen 5 1600X 40136
AMD Ryzen 5 2600X 39871
AMD Ryzen 5 1600 36536
Intel Core i5-9600K 35275
Intel Core i5-9400F 32863

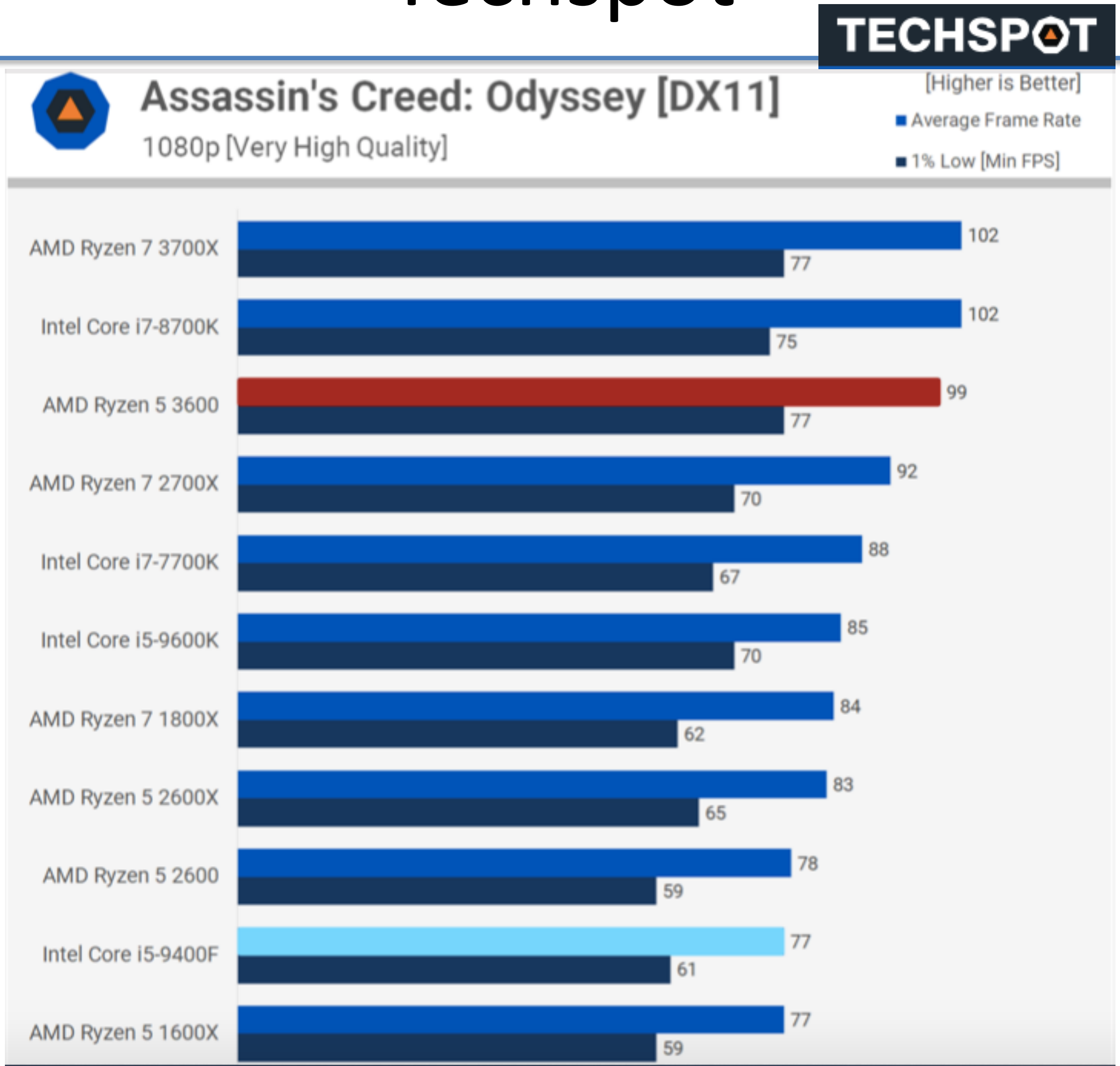
Techspot


TECHSPOT
Blender Open Data
Render Time
[Lower is Better]
Time in Seconds
AMD Ryzen 7 3700X 972
AMD Ryzen 7 2700X 1132
AMD Ryzen 7 1800X 1220
AMD Ryzen 5 3600 1338
Intel Core i7-8700K 1401
AMD Ryzen 5 2600X 1485
AMD Ryzen 5 2600 1593
AMD Ryzen 5 1600X 1616
AMD Ryzen 5 1600 1779
Intel Core i5-9600K 1891
Intel Core i7-7700K 1975

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Techspot

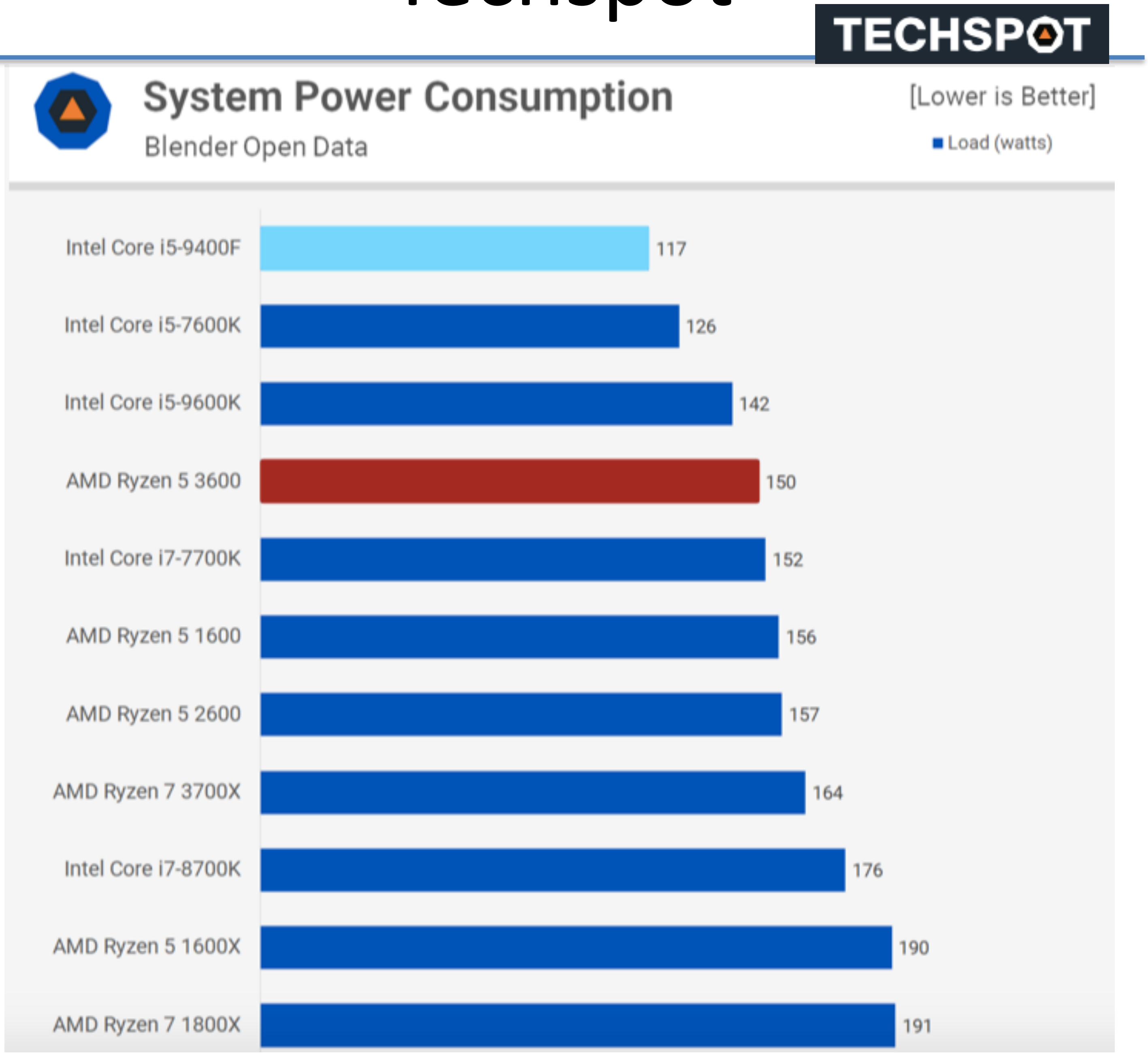
DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Techspot

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22



Section

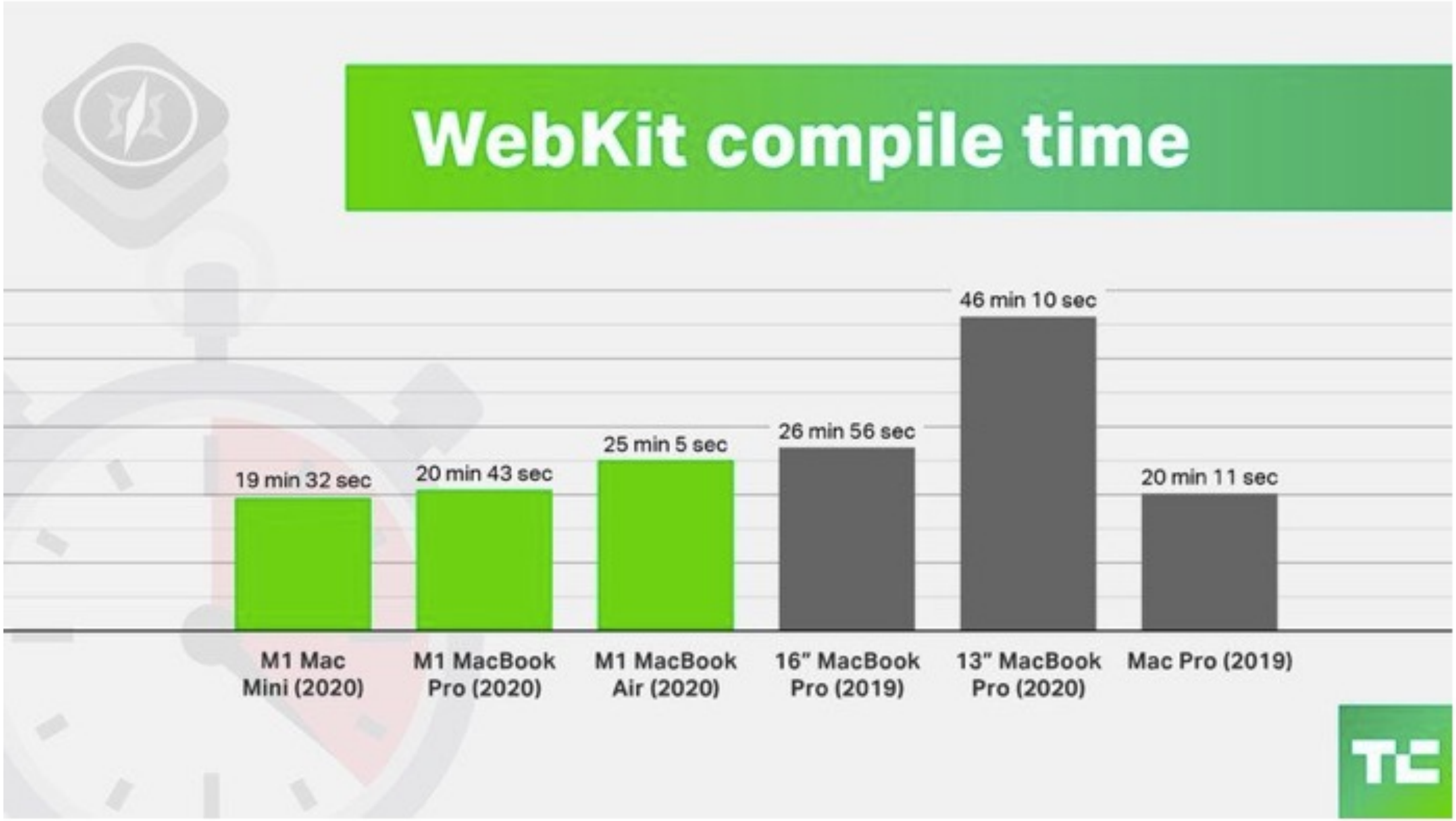
Apple M1 Benchmarks

M1 Benchmarks

CINEBENCH R23 SCORES

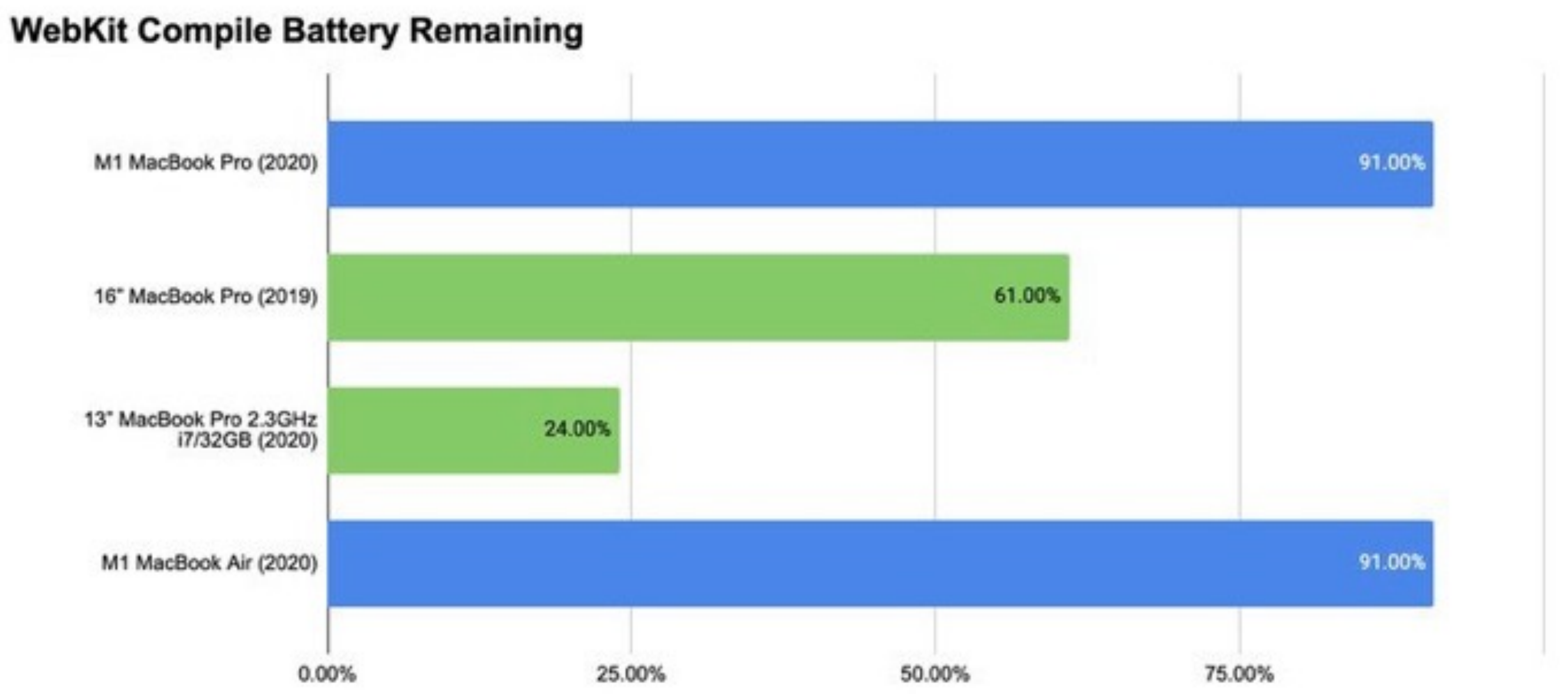
	16" M1 Max	14" M1 Pro (8 Core)	M1	2020 16" i7 Intel MBP
Single-Core	1521	1527	1519	1087
Multi-Core	12380	8773	7758	7313
30min Loop	12385	8772	7763	6877

Apple M1



As mentioned, things get really impressive when battery life is considered. After the WebKit compiling was finished on the various Macs, the M1-based MacBook Air and 13-inch MacBook Pro each had 91% battery life remaining, compared to 61% on a high-end 16-inch MacBook Pro and just 24% on the Intel-based 13-inch MacBook Pro.

Apple M1

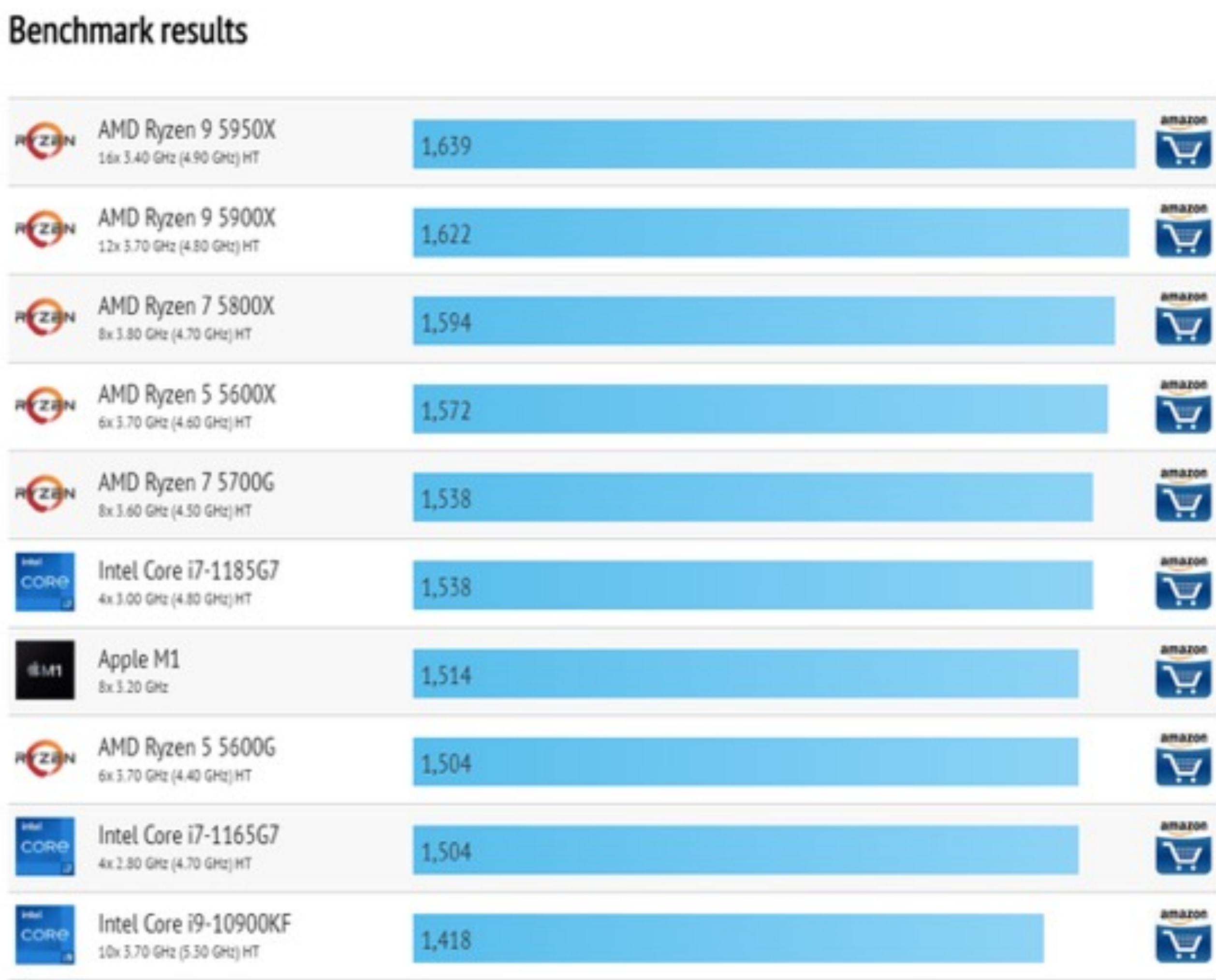


All in all, Apple's promise that its chips would deliver industry-leading performance-per-watt appears to be holding up. Panzarino's review has lots of other useful charts and benchmarks and is worth a read as customers wait for their new Macs to arrive.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Apple M1

DR JEFF SOFTWARE
INDIE APP DEVELOPER



The performance is still nothing to scoff at, considering how the M1 runs at <30 Watts while a Ryzen 5600G needs ***double*** the power with a pretty comparable result and a comparable Intel processor (i7-1165G7) is better, but still almost 30%

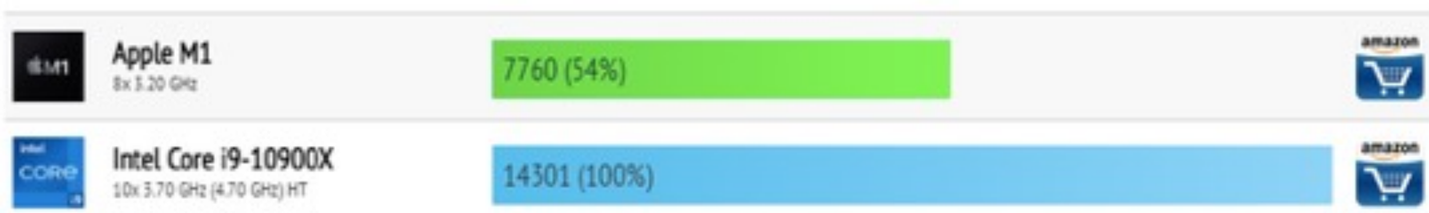
Apple M1

Cinebench R23 (Multi-Core)

Cinebench R23 is the successor of Cinebench R20 and is also based on the Cinema 4 Suite. Cinema 4 is a worldwide used software to create 3D forms. The multi-core test involves all CPU cores and taks a big advantage of hyperthreading.

Cinebench R23 (Multi-Core)

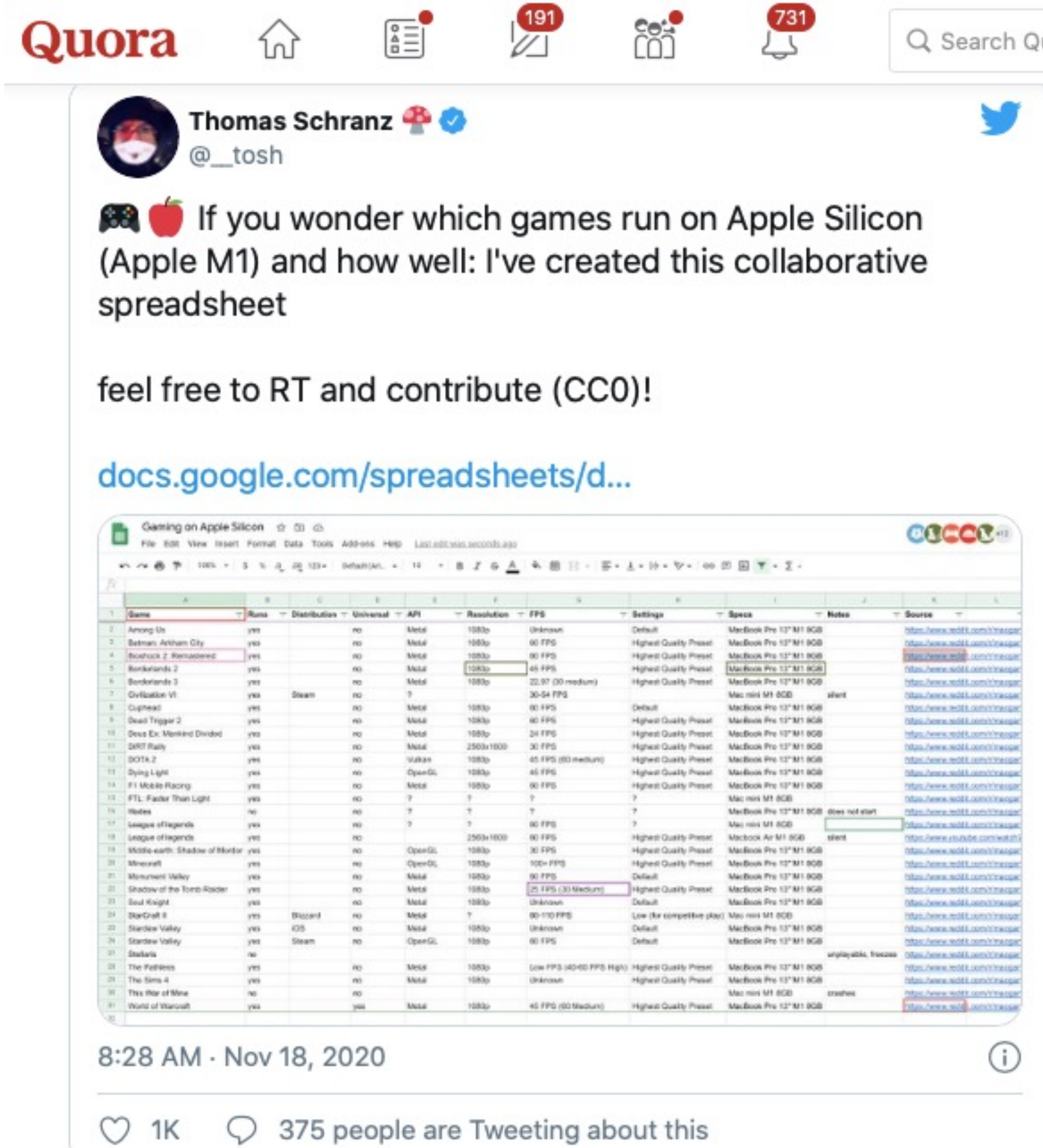
Cinebench R23 is the successor of Cinebench R20 and is also based on the Cinema 4 Suite. Cinema 4 is a worldwide used software to create 3D forms. The multi-core test involves all CPU cores and taks a big advantage of hyperthreading.



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Apple M1

DR JEFF SOFTWARE
INDIE APP DEVELOPER



Apple M1

Benchmarks

System	Uploaded	Platform	Single-Core Score	Multi-Core Score
MacBook Air (Late 2020) Apple M1 3196 MHz (8 cores)	December 6th, 2020	macOS	1742	7547

Geekbench.com creates benchmarks used to test a system's processing power. The newest MacBook air with the M1 chip tested at 1742 for single core and 7547 at multi core.

Kbytes	Inc32wds	Inc16wds	Inc8wds	Inc4wds	Inc2wds	ReadAll	128bSSE2	ReadAll	128bSSE2
6	31565	31291	31178	42042	42508	41978	61606	21375	61610
24	31300	31285	31258	42203	42786	41751	62331	21157	62329
96	5375	5559	5793	11083	20009	34332	40516	20363	40673
384	5562	5658	5864	11338	19966	33244	39317	20679	38413
768	5331	5391	5505	10966	19403	32805	37871	20680	38718
1536	5364	5427	5508	10779	19355	33166	37951	20679	38331
16380	1070	1356	1955	4248	8046	16688	16838	14103	16757
131070	1034	1272	1866	4023	7724	16029	15980	13852	15963

Part 2 - 2 Thread MBytes/Second

Kbytes	Inc32wds	Inc16wds	Inc8wds	Inc4wds	Inc2wds	ReadAll	128bSSE2	ReadAll	128bSSE2
6	63147	62371	62552	83983	85074	83689	123233	42597	123206
24	62579	62580	62188	84353	85351	83515	124250	42252	124188
96	10779	10875	11473	21904	39332	67550	80624	40717	80597
384	10088	11391	11560	22649	39705	67022	78033	41352	76206
768	10574	10610	11042	21889	38669	65967	76066	41356	77275
1536	10442	10637	10901	21597	38467	66046	75829	41353	76302
16380	1798	2305	3397	7161	13013	28647	28743	25080	28471

Apple M1

Single-Core Performance

	Mac mini (Late 2020)	MacPro7,1
Single-Core Score	**1732**	**2119**
AES-XTS	2604 4.44 GB/sec	3691 6.29 GB/sec
Text Compression	1431 7.24 MB/sec	2228 11.3 MB/sec
Image Compression	1375 65.0 Mpixels/sec	1918 90.7 Mpixels/sec
Navigation	1761 4.97 MTE/sec	1851 5.22 MTE/sec
HTML5	1679 1.97 MElements/sec	1624 1.91 MElements/sec
SQLite	1422 445.4 Krows/sec	1748 547.6 Krows/sec
PDF Rendering	1605 87.1 Mpixels/sec	1978 107.3 Mpixels/sec
Text Rendering	1801 573.7 KB/sec	1743 555.5 KB/sec

Apple M1

	Apple iPad pro 12.9 (M1)	MacBook Air M1	Microsoft Surface Pro 7 (Intel Core i5-1035G4 10th Gen)
Single-Core	**1714**	1740	862
Multi-Core	**7272**	7694	3104
OpenCL Score	**20887**	18347	7801

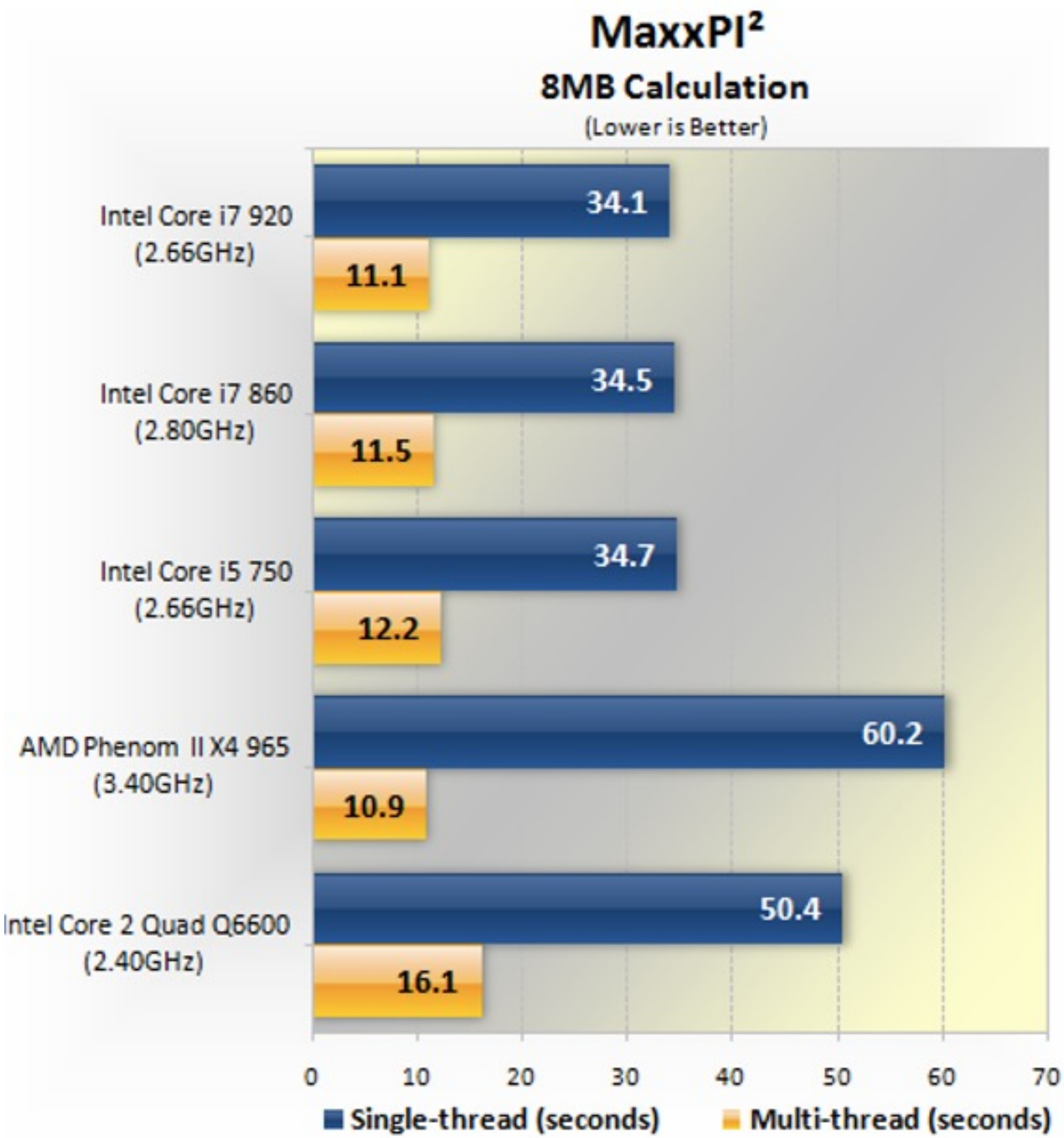
Apple's M1-based iPad Pro's benchmarks are in line with other M1-based systems.

Section

Misc Benchmarks

COMP222

Misc Benchmarks



Misc Benchmarks

CPU Gaming Benchmark Hierarchy Post-Zen 3

	1080p Gaming Score	1440p Gaming Score	CPU	Cores/Threads	Base/Boost	TDP	Buy
Ryzen 9 5900X	100%	100%	Zen 3	12/24	3.7 / 4.8 GHz	105W	
Ryzen 9 5950X	99.77%	99.38%	Zen 3	16/32	3.4 / 4.9 GHz	105W	
Ryzen 7 5800X	97.22%	99%	Zen 3	8/16	3.8 / 4.7 GHz	105W	
Ryzen 5 5600X	96.90%	95.30%	Zen 3	6/12	3.7 / 4.6 GHz	65W	
Intel Core i9-10900K	88.97%	95.30%	Comet Lake	10/20	3.7 / 5.3 GHz	125W	
Intel Core i9-10850K	87.36%	94.52%	Comet Lake	10/20	3.6 / 5.2 GHz	95W	
Core i7-10700K	84.39%	92.05%	Comet Lake	8/16	3.8 / 5.1 GHz	125W	
Intel Core i9-10980XE	83.64%	88.18%	Cascade Lake-X	18/36	3.0 / 4.8 GHz	165W	

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Misc Benchmarks

AIDA64 GPGPU Benchmark

☑ GPU: nVIDIA GeForce GTX 1080 (GP104)
1822 MHz, 2560 cores, 20 CUs, Driver 457.30

☑ CPU: Intel Core i5-9600K (Coffee Lake-S)
4533 MHz, 6 cores, 6 threads

	GPU	x64 CPU
Memory Read	12705 MB/s	[TRIAL VERSION]
Memory Write	[TRIAL VERSION]	43777 MB/s
Memory Copy	228347 MB/s	37164 MB/s
Single-Precision FLOPS	9286 GFLOPS	[TRIAL VERSION]
Double-Precision FLOPS	306.1 GFLOPS	418.1 GFLOPS
24-bit Integer IOPS	3217 GIOPS	371.3 GIOPS
32-bit Integer IOPS	[TRIAL VERSION]	371.5 GIOPS
64-bit Integer IOPS	526.3 GIOPS	52.29 GIOPS
AES-256	26022 MB/s	28882 MB/s
SHA-1 Hash	51292 MB/s	[TRIAL VERSION]
Single-Precision Julia	1853 FPS	[TRIAL VERSION]
Double-Precision Mandel	[TRIAL VERSION]	147.1 FPS

AIDA64 v6.30.5500 (c) 1995-2020 FinalWire Ltd.

Misc Benchmarks

Intel Core i7-3770K @ 3.50GHz		Average CPU Mark
Class: Desktop	**Socket:** LGA1155	6443
Clockspeed: 3.5 GHz	**Turbo Speed:** 3.9 GHz	
Cores: 4 **Threads:** 8	**Typical TDP:** 77 W	
		Single Thread Rating: 2075
Other names: Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz		Cross-Platform Rating: 13,885
CPU First Seen on Charts: Q2 2012		Samples: 8491*
CPUmark/$Price: 30.79		*Margin for error: Low
Overall Rank: 703		+ COMPARE
Last Price Change: $209.27 USD (2020-12-05)		**PerformanceTest V9** CPU Mark: 9,370 Thread: 2,077

Misc Benchmarks

Intel Core i7-9700KF @ 3.60GHz		Average CPU Mark
Class: Desktop	**Socket:** FCLGA1151-2	14725
Clockspeed: 3.6 GHz	**Turbo Speed:** 4.9 GHz	
Cores: 8 **Threads:** 8	**Typical TDP:** 95 W	
Other names: Intel(R) Core(TM) i7-9700KF CPU @ 3.60GHz		Single Thread Rating: 2948
CPU First Seen on Charts: Q2 2019		Cross-Platform Rating: 27,499
CPUmark/$Price: 49.25		Samples: 308*
Overall Rank: 238		*Margin for error: Low
Last Price Change: $299.00 USD (2020-12-04)		+ COMPARE
		PerformanceTest V9 CPU Mark: 16,975 Thread: 2,793

Misc Benchmarks

CPU

Architecture	1x 3.1 GHz – Kryo 585 Prime (Cortex-A77) 3x 2.42 GHz – Kryo 585 Gold (Cortex-A77) 4x 1.8 GHz – Kryo 585 Silver (Cortex-A55)	2x 3.1 GHz – Lightning 4x 1.8 GHz – Thunder
Cores	8	6
Frequency	3100 MHz	3100 MHz
Instruction set	ARMv8.2-A	ARMv8.4-A
L1 cache	512 KB	-
L2 cache	1 MB	-
L3 cache	4 MB	-
Process	7 nanometers	5 nanometers
Transistor count	-	11.8 billion
TDP	10 W	6 W

Misc Benchmarks

Graphics

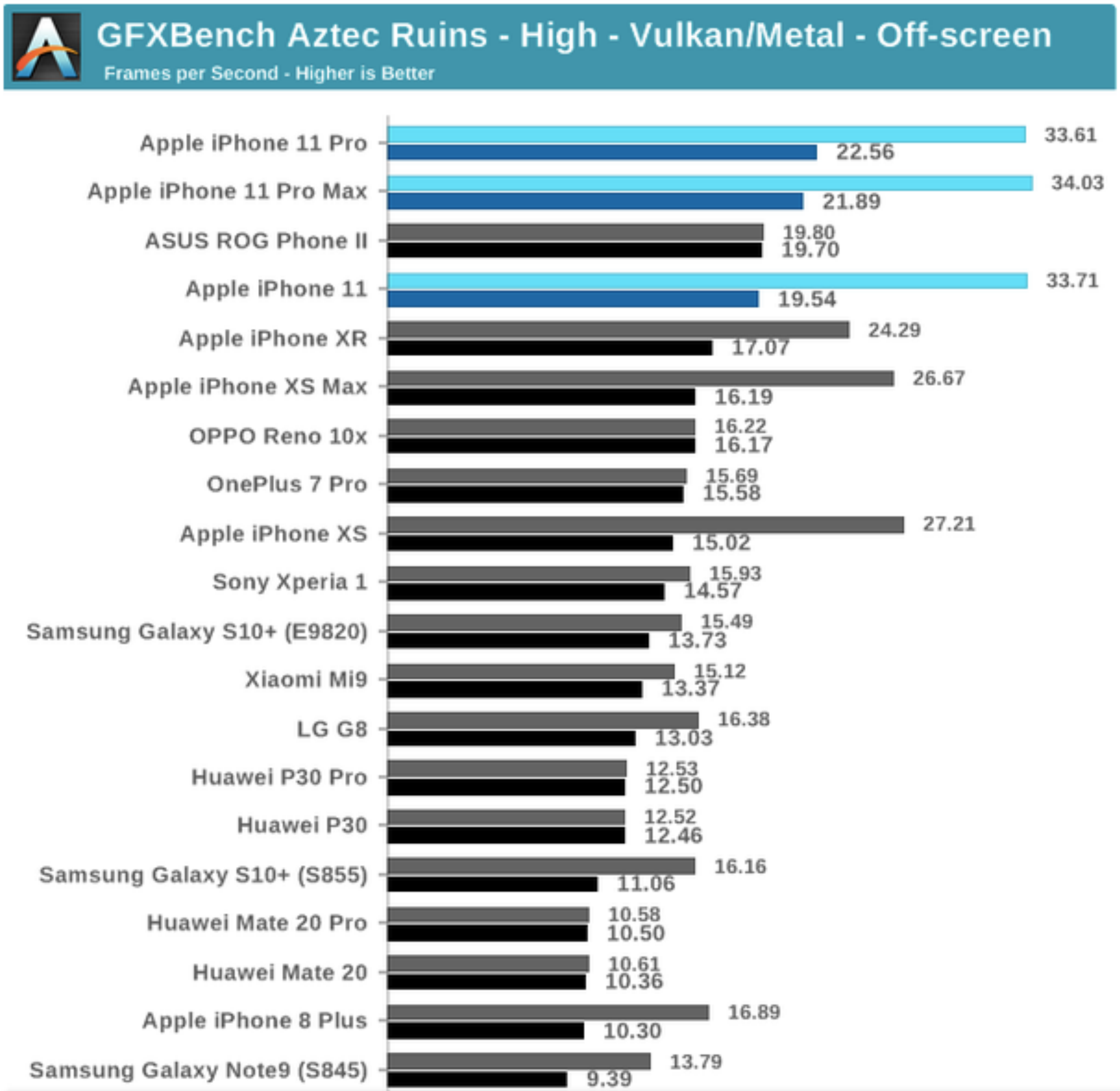
GPU name	Adreno 650	Apple GPU
Architecture	Adreno 600	-
GPU frequency	645 MHz	-
Execution units	2	4
Shading units	512	-
FLOPS	1365 Gigaflops	-
Vulkan version	1.1	-
OpenCL version	2.0	-
DirectX version	12	-

Section

Graphics (GPU) Benchmarks

CPU Performance

Frames/sec

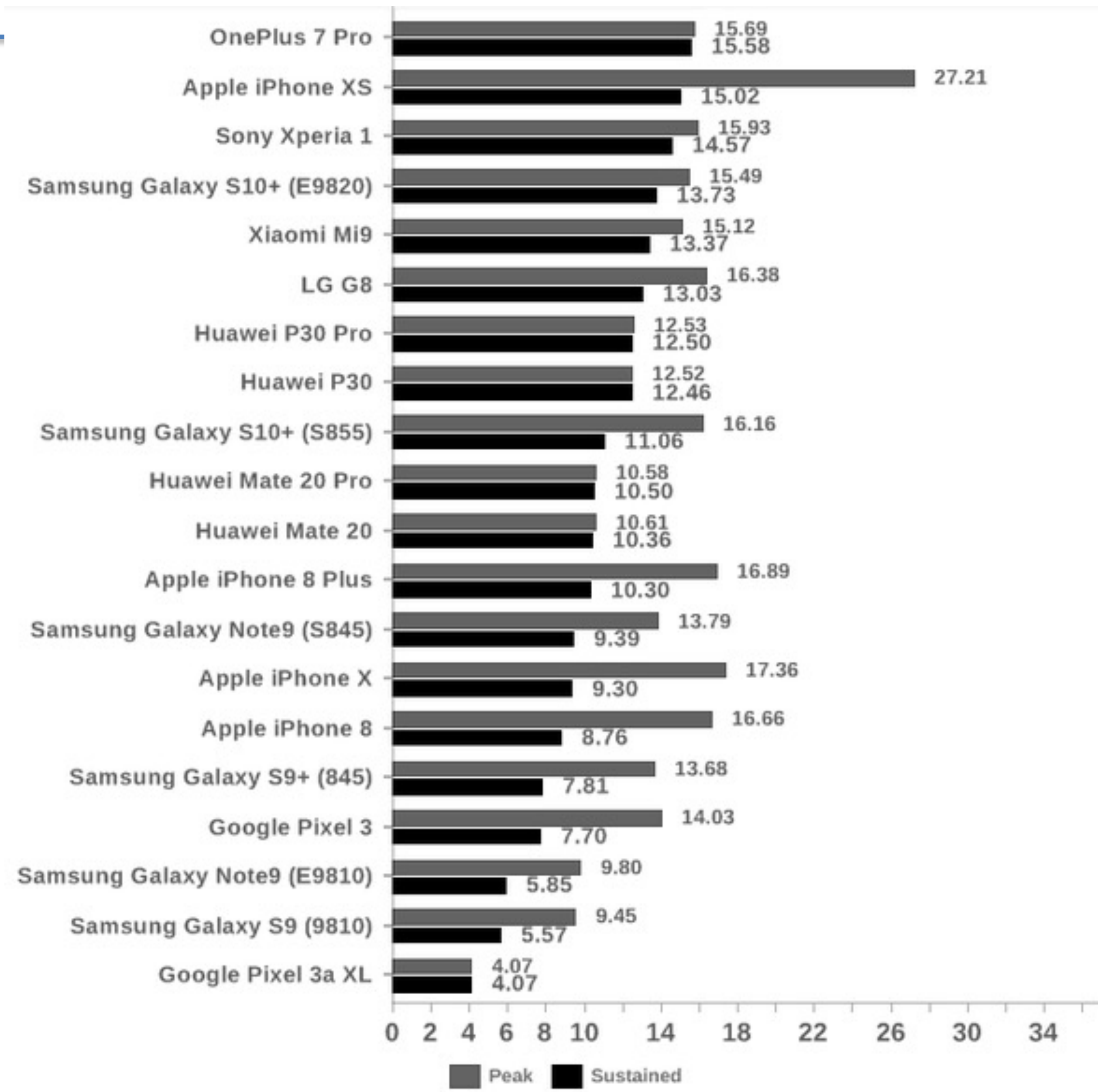


CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

CPU Performance

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Frames/sec



Section

Gaming Benchmarks

Gaming

Quora

Search Quora

Game	Runs	Distribution	Universal	API	Resolution	FPS
Among Us	yes		no	Metal	1080p	Unknown
Batman: Arkham City	yes		no	Metal	1080p	60 FPS
Bioshock 2: Remastered	yes		no	Metal	1080p	60 FPS
Borderlands 2	yes		no	Metal	1080p	45 FPS
Borderlands 3	yes		no	Metal	1080p	22.97 (30 medium)
Civilization VI	yes	Steam	no	?		30-54 FPS
Cuphead	yes		no	Metal	1080p	60 FPS
Dead Trigger 2	yes		no	Metal	1080p	60 FPS
Deus Ex: Mankind Divided	yes		no	Metal	1080p	24 FPS
DiRT Rally	yes		no	Metal	2560x1600	30 FPS
DOTA 2	yes		no	Vulkan	1080p	45 FPS (60 medium)
Dying Light	yes		no	OpenGL	1080p	45 FPS
F1 Mobile Racing	yes		no	Metal	1080p	60 FPS
FTL: Faster Than Light	yes		no	?	?	?
Hades	no		no	?	?	?
League of legends	yes		no	?	?	60 FPS
League of legends	yes		no		2560x1600	60 FPS
Middle-earth: Shadow of Mordor	yes		no	OpenGL	1080p	30 FPS
Minecraft	yes		no	OpenGL	1080p	100+ FPS
Monument Valley	yes		no	Metal	1080p	60 FPS
Shadow of the Tomb Raider	yes		no	Metal	1080p	25 FPS (30 Medium)
Soul Knight	yes		no	Metal	1080p	Unknown
StarCraft II	yes	Blizzard	no	Metal	?	60-110 FPS
Stardew Valley	yes	iOS	no	Metal	1080p	Unknown
Stardew Valley	yes	Steam	no	OpenGL	1080p	60 FPS
Stellaris	no					
The Pathless	yes		no	Metal	1080p	Low FPS (40-60 FPS High)
The Sims 4	yes		no	Metal	1080p	Unknown
This War of Mine	no		no			
World of Warcraft	yes		yes	Metal	1080p	45 FPS (60 Medium)

Gaming

Quora | Search Quora

Some people are OK with less than 30 FPS, but I would say that in action games (which all of the above are) anything less than 30 is unsatisfactory. Less than 20 is unplayable. No matter how you look at it, nobody would ever call sub-30 FPS 'high.'

It is not a bad showing for integrated graphics, but this is still a very long way from calling a machine with the M1 a 'gaming PC.' Further, to be a gaming PC you *kind of need to be able to run games on it*. There's a reason there aren't very many entries on that list yet; it's because the vast majority of released games won't even run on an M1 Mac.

Interesting note: There were many synthetic benchmarks claiming that the M1's iGPU crushed the 4-year-old GTX 1050 Ti discrete GPU. However, these results are interesting, especially since those games are running natively.

NVIDIA GeForce GTX 1050 Ti (Notebook) - NotebookCheck.net Tech

Deus Ex: Mankind Divided: **39 FPS** at 1080p (high)

Shadow of the Tomb Raider: **35 FPS** at 1080p (high)

Borderlands 3: **28 FPS** at 1080p (high)

That's *considerably better* performance across the board - actually playable in most cases. This shows that those synthetic benchmarks do not reflect actual usage properly, or there are other factors (Drivers? DirectX vs. Metal?) that are affecting results dramatically.

Section

Mobile Benchmarks

Mobile Benchmarks

NANOREVIEW.NET Smartphone Processors CPU Comparison

Home > Smartphone processors comparison > Snapdragon 865 Plus vs A14 Bionic – what's better?

Snapdragon 865 Plus vs A14 Bionic

Qualcomm snapdragon VS Apple A Series

Snapdragon 865 Plus | A14 Bionic

We compared the 8-core Qualcomm Snapdragon 865 Plus (Adreno 650) with the newer 6-core Apple A14 Bionic (Apple GPU) SoC. Here you will find the pros and cons of each chip, technical specs, and comprehensive tests in benchmarks, like AnTuTu and Geekbench.

Snapdragon

Review

General comparison of performance, power consumption, and other indicators

CPU Performance

Single and multi-core processor tests

- Snapdragon 865 Plus: 83
- A14 Bionic: 96

Gaming Performance

GPU performance in games and OpenCL/Vulcan

- Snapdragon 865 Plus: 97
- A14 Bionic: 100

Battery life

Efficiency of battery consumption

- Snapdragon 865 Plus: 82
- A14 Bionic: 94

NanoReview Score

Overall chip score

- Snapdragon 865 Plus: 88
- A14 Bionic: 97

Mobile

Phone	Processor	Geekbench 5 single-core result	Geekbench 5 multicore result
iPhone 12	A14 Bionic	1,593	3,859
iPhone 12 Pro	A14 Bionic	1,595	3,880
iPhone 11 Pro Max	A13 Bionic	1,334	3,517
Samsung Galaxy Note 20 Ultra	Snapdragon 865 Plus	985	3,294
Samsung Galaxy S20 Plus	Snapdragon 865	811	3,076
OnePlus 8T	Snapdragon 865	887	3,203

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Benchmarks

DR JEFF SOFTWARE INDIE APP DEVELOPER

Benchmarks

6	iPhone xs	47690
7	iPhone xr	47555
8	iPhone xs max	47343
9	iPhone 8 plus	10170
10	iPhone x	9869
11	iPhone 8	9693
12	iPhone 7 plus	9176
13	iPhone 7	9103
14	iPhone 6s plus	7629
15	iPhone 6s	7586

Benchmarks

Geekbench Browser | Geekbench 5 | Geekbench 4 | Benchmark Charts | Search

Samsung Galaxy S20+ 5G Benchmarks

Benchmark results for the Samsung Galaxy S20+ 5G can be found below. The data on this chart is gathered from user-submitted Geekbench 5 results from the Geekbench Browser.

Geekbench 5 scores are calibrated against a baseline score of 1000 (which is the score of an Intel Core i3-8100). Higher scores are better, with double the score indicating double the performance.

CPU Benchmark Scores

Single-Core Score	Multi-Core Score
827	3100

Compute Benchmark Scores

Vulkan Score	OpenCL Score
N/A	3170

Benchmarks

Geekbench Browser | Geekbench 5 | Geekbench 4 | Benchmark Charts | Search

Device Information

Name	Samsung Galaxy S20+ 5G
Processor	Qualcomm Snapdragon 865
Processor Frequency	1804 MHz
Processor Cores	8

Samsung Benchmarks

Single-Core | Multi-Core | OpenCL

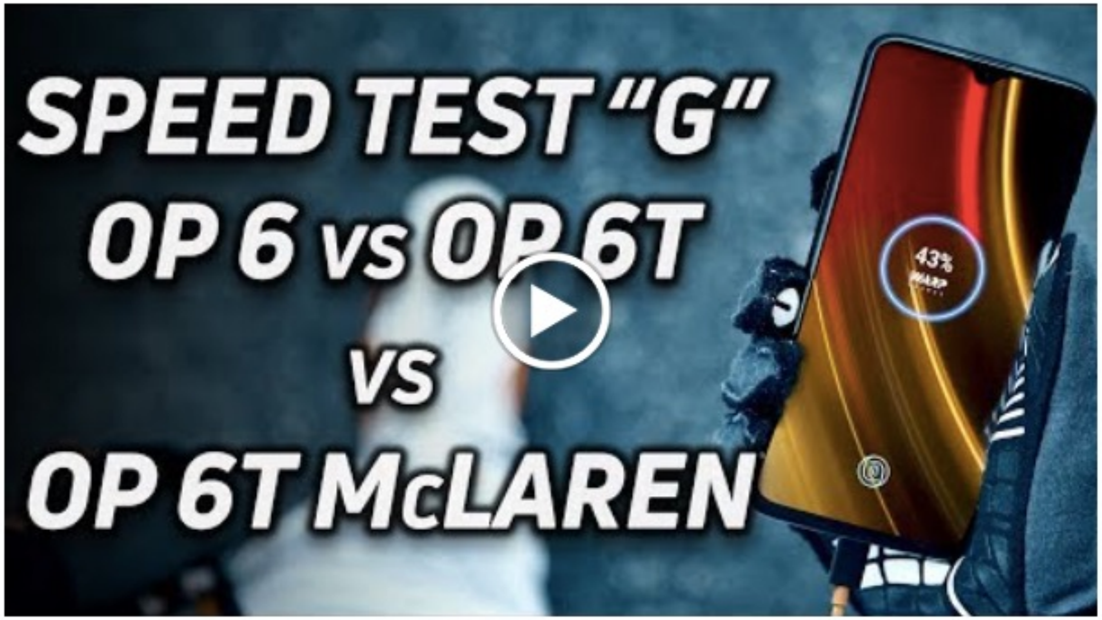
Processor	Score
Samsung Galaxy S20 Ultra 5G Qualcomm Snapdragon 865 @ 1.8 GHz	845
Samsung Galaxy S20+ 5G Qualcomm Snapdragon 865 @ 1.8 GHz	827
Samsung Galaxy S20 5G Qualcomm Snapdragon 865 @ 1.8 GHz	827
Samsung Galaxy S20 Ultra Samsung Exynos 990 @ 2.0 GHz	805
Samsung Galaxy S20+ Samsung Exynos 990 @ 2.0 GHz	796
Samsung Galaxy S20 Samsung Exynos 990 @ 2.0 GHz	765

Benchmarks

Quora | 1 Home | Following | 209 Answer | Spaces | 732 Notifications

- All Android smartphone manufacturers, ***bar none***, have been caught cheating on all benchmarks, because all Android SoCs are much slower than Apple SoCs
- Read and watch ***all*** the proof:
- Phones we caught cheating benchmarks in 2018

Samsung owes Galaxy S4 owners $10 for cheating on benchmarks



Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated

They're (Almost) All Dirty: The State of Cheating in Android Benchmarks

Huawei & Honor's Recent Benchmarking Behaviour: A Cheating Headache

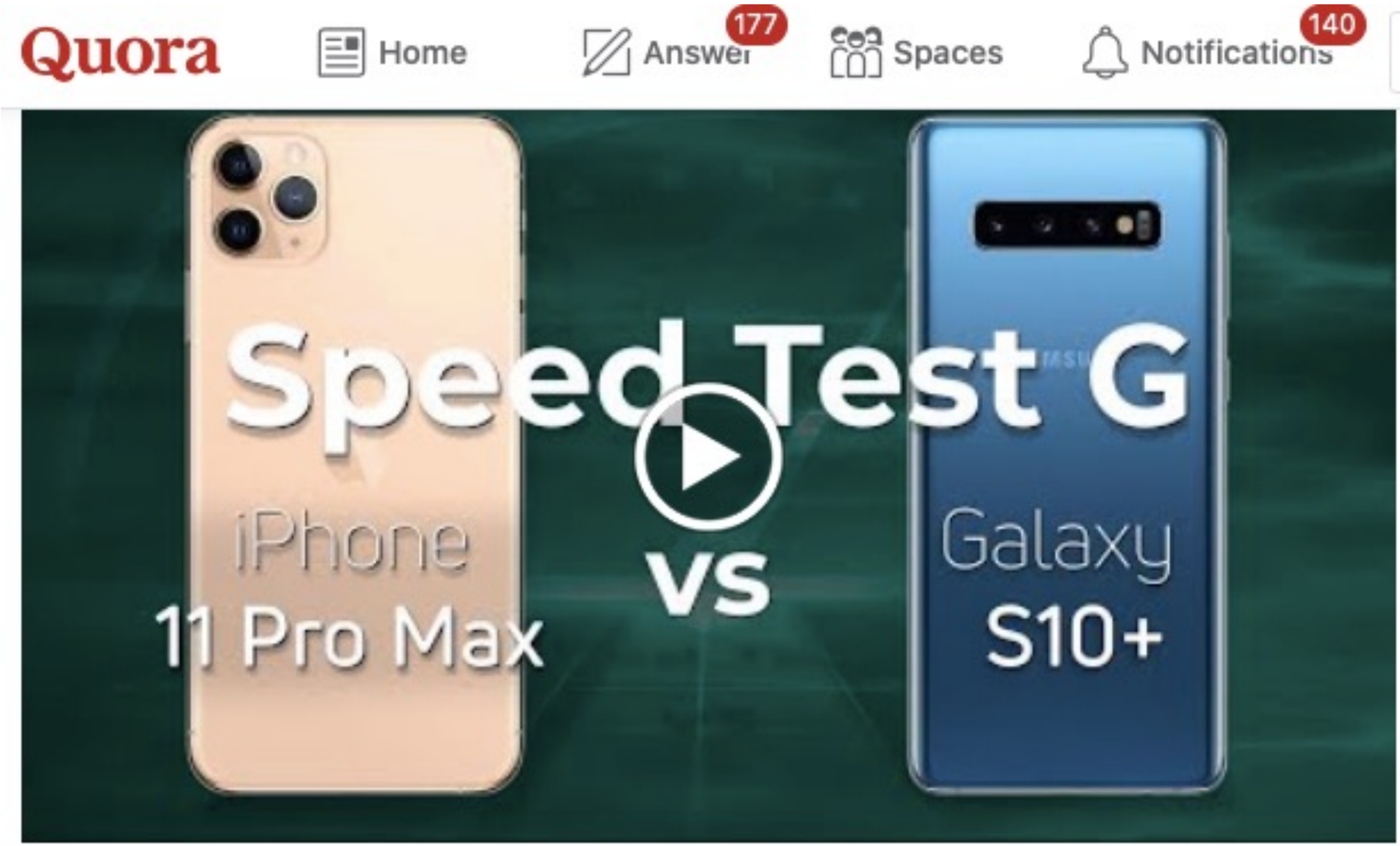
Benchmarks

Quora | Home 1 | Following | Answer 209 | Spaces | Notifications 732

- Huawei & Honor's Recent Benchmarking Behaviour: A Cheating Headache
- Why and how do OEMs cheat on benchmarking? - Gary explains
- Almost all Android devices cheat at benchmarks, report says
- Android manufacturers just can't stop cheating on benchmark tests
- Popular Android manufacturer OnePlus caught cheating in benchmark tests
- Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated
- OnePlus 5 cheats in benchmarks to get high scores
- OnePlus 5 caught cheating on multi-core benchmarks
- Huawei/Honor caught cheating in Benchmarks! And a shameless justification.
- Samsung again cheats Benchmark scores, this time with the Galaxy Note 3
- https://www.androidpit.com/huawei-cheats-in-benchmarks-tests
- Oppo F7 caught red-handed cheating on benchmarks by artificially boosting performance in certain apps- Technology News, Firstpost
- Geenkbench and Antutu Benchmark Scores: All You Need to Know | Suggest Phone

Benchmarks: Mobile

Gary's Youtube channel
https://www.youtube.com/watch?time_continue=13&v=nvbjAhM6K1I&feature=emb_logo



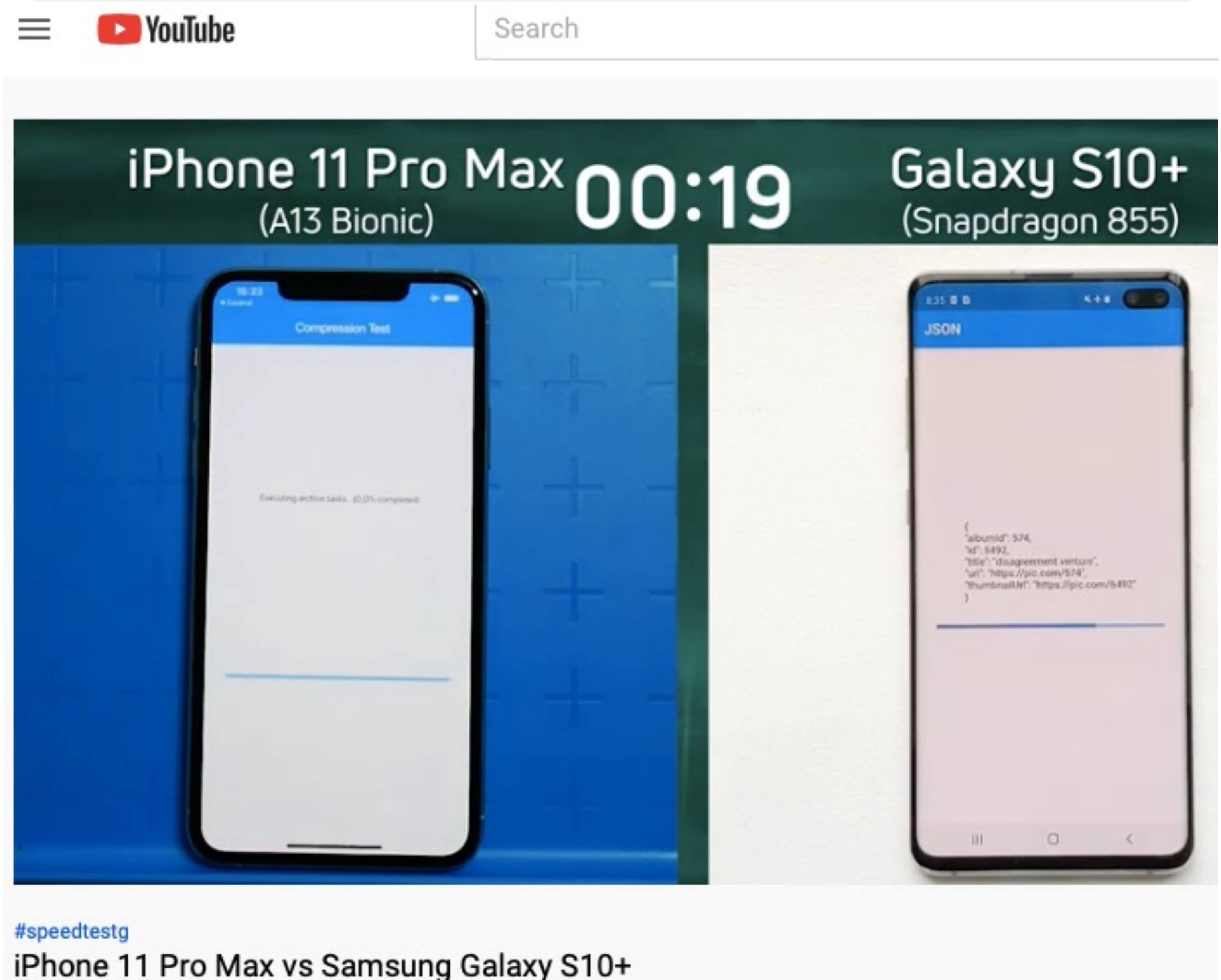
Benchmarks: Mobile

Speed Test G

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Benchmarks: Mobile

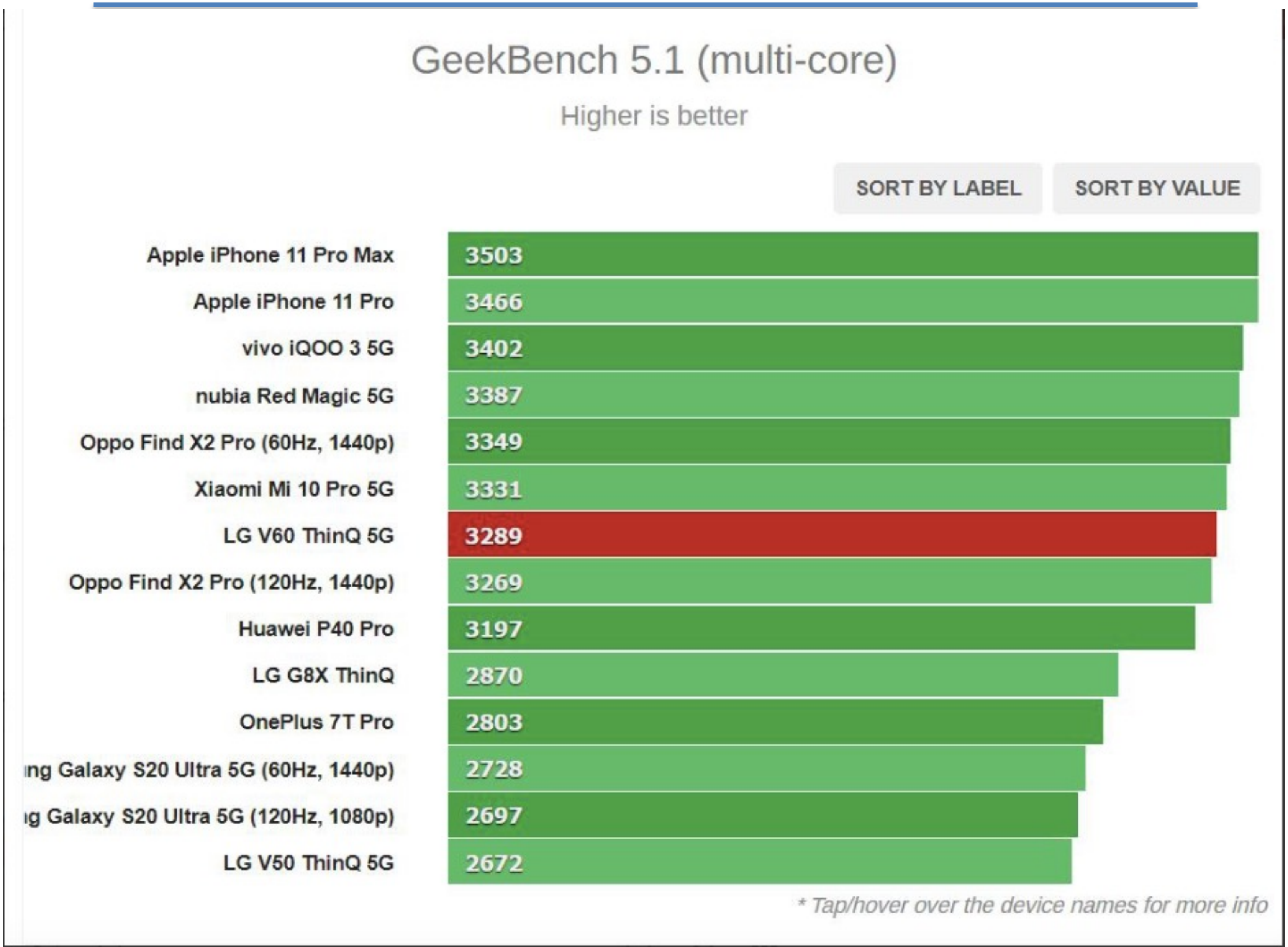
DR JEFF SOFTWARE
INDIE APP DEVELOPER



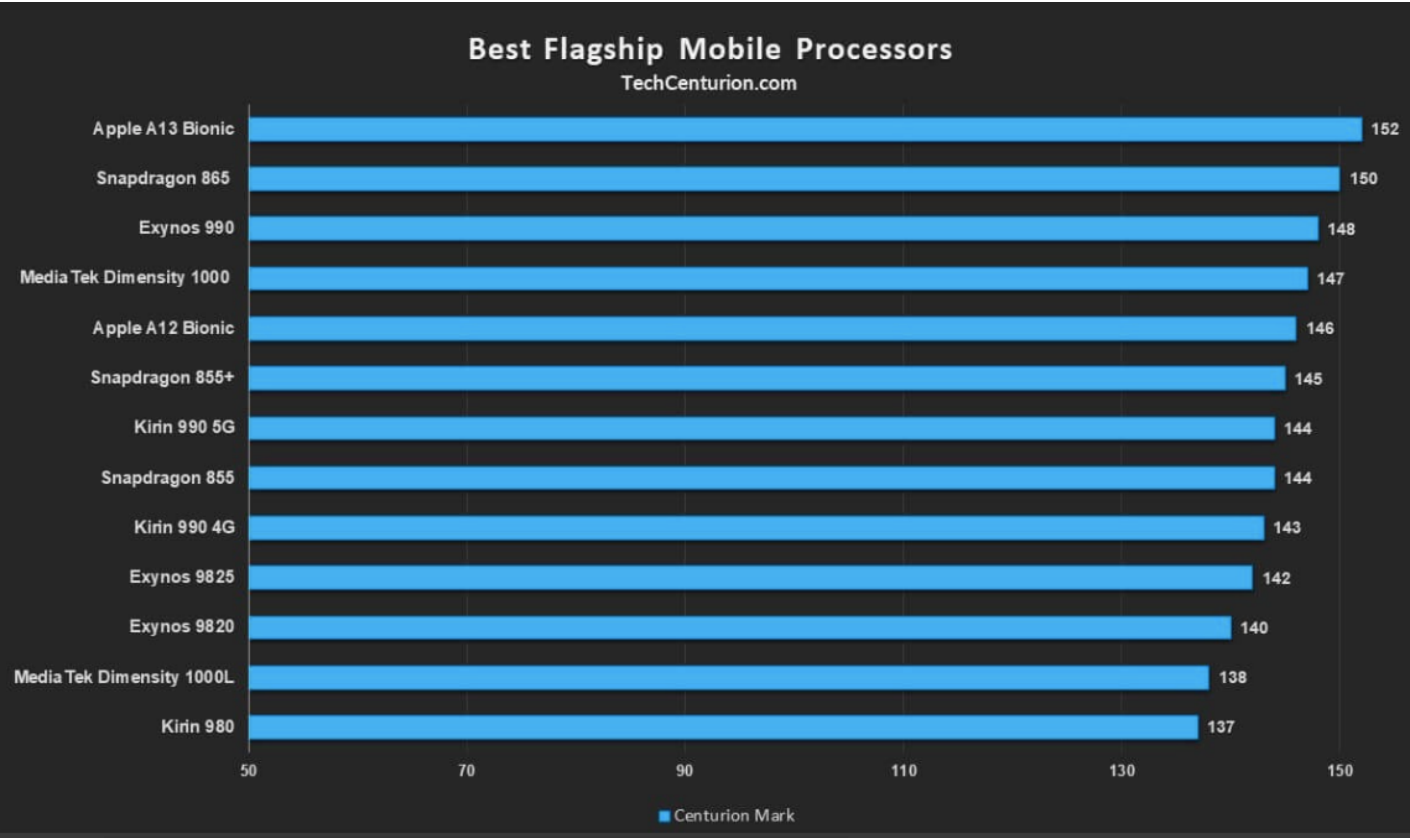
Benchmarks: Mobile

	iPhone 11 Pro Max	Galaxy S10+
CPU	40.0	48.0
MIXED	21.0	27.3
GPU	14.5	22.4
TOTAL	1:15.5	1:37.8

Benchmarks: Mobile



Benchmarks: Mobile



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Benchmarks

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2020-22

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

Benchmarks

6	iPhone xs	47690
7	iPhone xr	47555
8	iPhone xs max	47343
9	iPhone 8 plus	10170
10	iPhone x	9869
11	iPhone 8	9693
12	iPhone 7 plus	9176
13	iPhone 7	9103
14	iPhone 6s plus	7629
15	iPhone 6s	7586

Benchmarks

Geekbench Browser Geekbench 5 Geekbench 4 Benchmark Charts Search

Samsung Galaxy S20+ 5G Benchmarks

Benchmark results for the Samsung Galaxy S20+ 5G can be found below. The data on this chart is gathered from user-submitted Geekbench 5 results from the Geekbench Browser.

Geekbench 5 scores are calibrated against a baseline score of 1000 (which is the score of an Intel Core i3-8100). Higher scores are better, with double the score indicating double the performance.

CPU Benchmark Scores

Single-Core Score	Multi-Core Score
827	3100

Compute Benchmark Scores

Vulkan Score	OpenCL Score
N/A	3170

Benchmarks

Geekbench Browser | Geekbench 5 | Geekbench 4 | Benchmark Charts | Search

Device Information

Name	Samsung Galaxy S20+ 5G
Processor	Qualcomm Snapdragon 865
Processor Frequency	1804 MHz
Processor Cores	8

Samsung Benchmarks

Single-Core | Multi-Core | OpenCL

Processor	Score
Samsung Galaxy S20 Ultra 5G Qualcomm Snapdragon 865 @ 1.8 GHz	845
Samsung Galaxy S20+ 5G Qualcomm Snapdragon 865 @ 1.8 GHz	827
Samsung Galaxy S20 5G Qualcomm Snapdragon 865 @ 1.8 GHz	827
Samsung Galaxy S20 Ultra Samsung Exynos 990 @ 2.0 GHz	805
Samsung Galaxy S20+ Samsung Exynos 990 @ 2.0 GHz	796
Samsung Galaxy S20 Samsung Exynos 990 @ 2.0 GHz	765

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

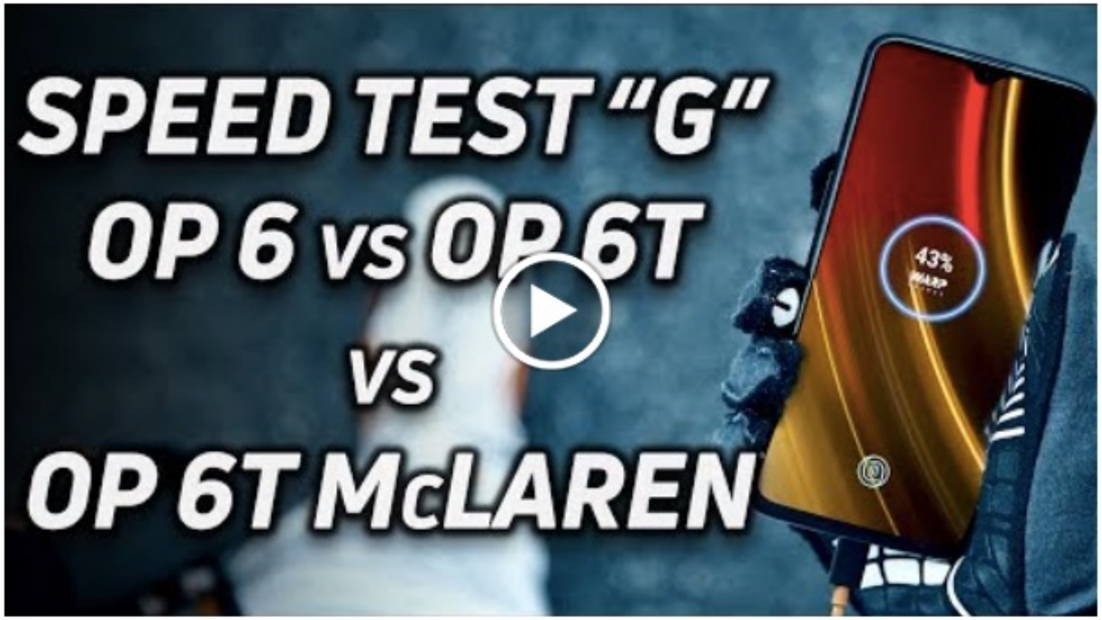
Benchmarks

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Quora Home Following Answer Spaces Notifications

- All Android smartphone manufacturers, ***bar none***, have been caught cheating on all benchmarks, because all Android SoCs are much slower than Apple SoCs
- Read and watch ***all*** the proof:
- Phones we caught cheating benchmarks in 2018

Samsung owes Galaxy S4 owners $10 for cheating on benchmarks



Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated

They're (Almost) All Dirty: The State of Cheating in Android Benchmarks

Huawei & Honor's Recent Benchmarking Behaviour: A Cheating Headache

Benchmarks

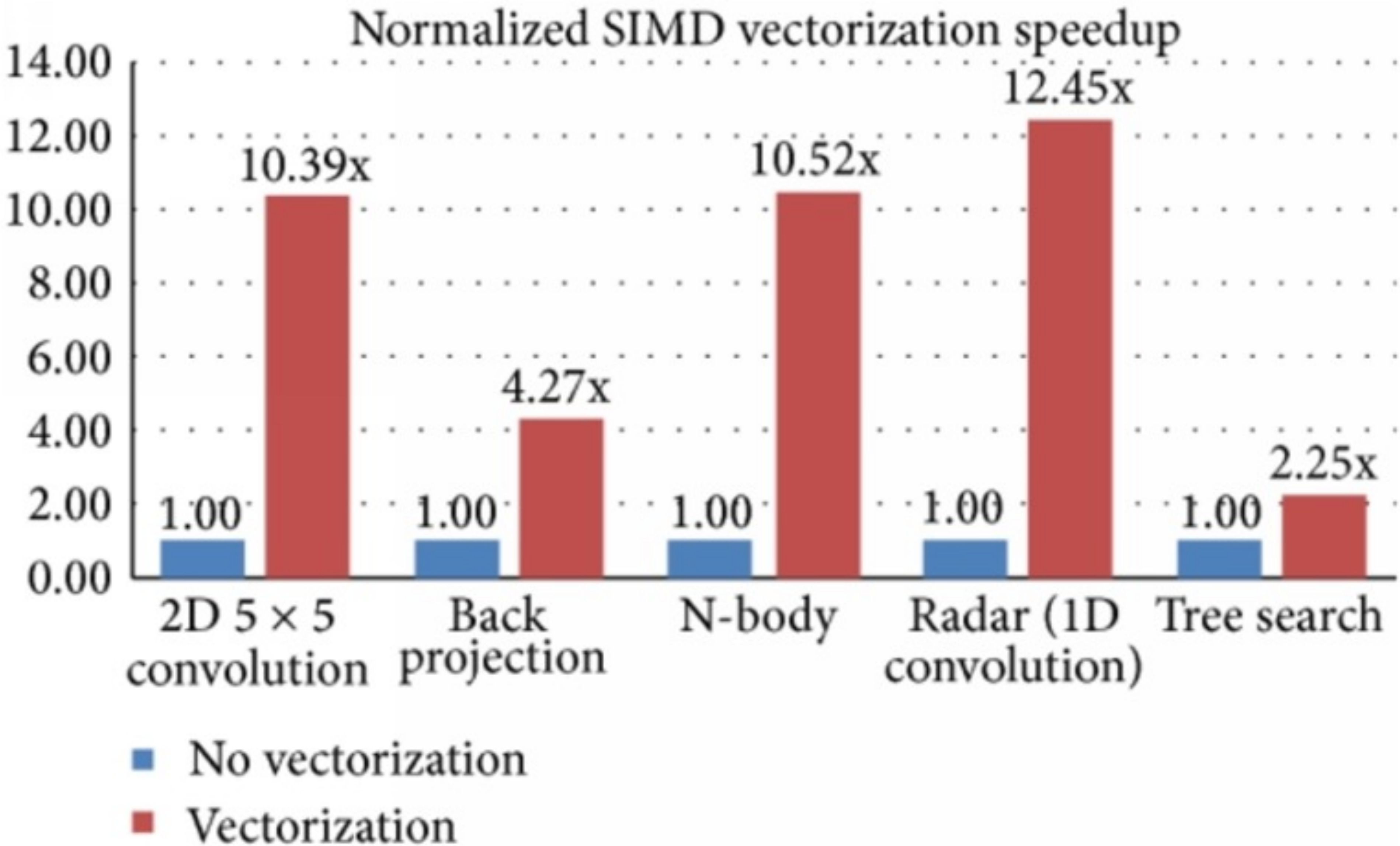
Quora | 1 Home | Following | 209 Answer | Spaces | 732 Notifications

- Huawei & Honor's Recent Benchmarking Behaviour: A Cheating Headache
- Why and how do OEMs cheat on benchmarking? - Gary explains
- Almost all Android devices cheat at benchmarks, report says
- Android manufacturers just can't stop cheating on benchmark tests
- Popular Android manufacturer OnePlus caught cheating in benchmark tests
- Do NOT Trust OnePlus 5 Benchmarks in Reviews - How OnePlus Cheated
- OnePlus 5 cheats in benchmarks to get high scores
- OnePlus 5 caught cheating on multi-core benchmarks
- Huawei/Honor caught cheating in Benchmarks! And a shameless justification.
- Samsung again cheats Benchmark scores, this time with the Galaxy Note 3
- https://www.androidpit.com/huawei-cheats-in-benchmarks-tests
- Oppo F7 caught red-handed cheating on benchmarks by artificially boosting performance in certain apps- Technology News, Firstpost
- Geenkbench and Antutu Benchmark Scores: All You Need to Know | Suggest Phone

Section

SIMD
Benchmarks

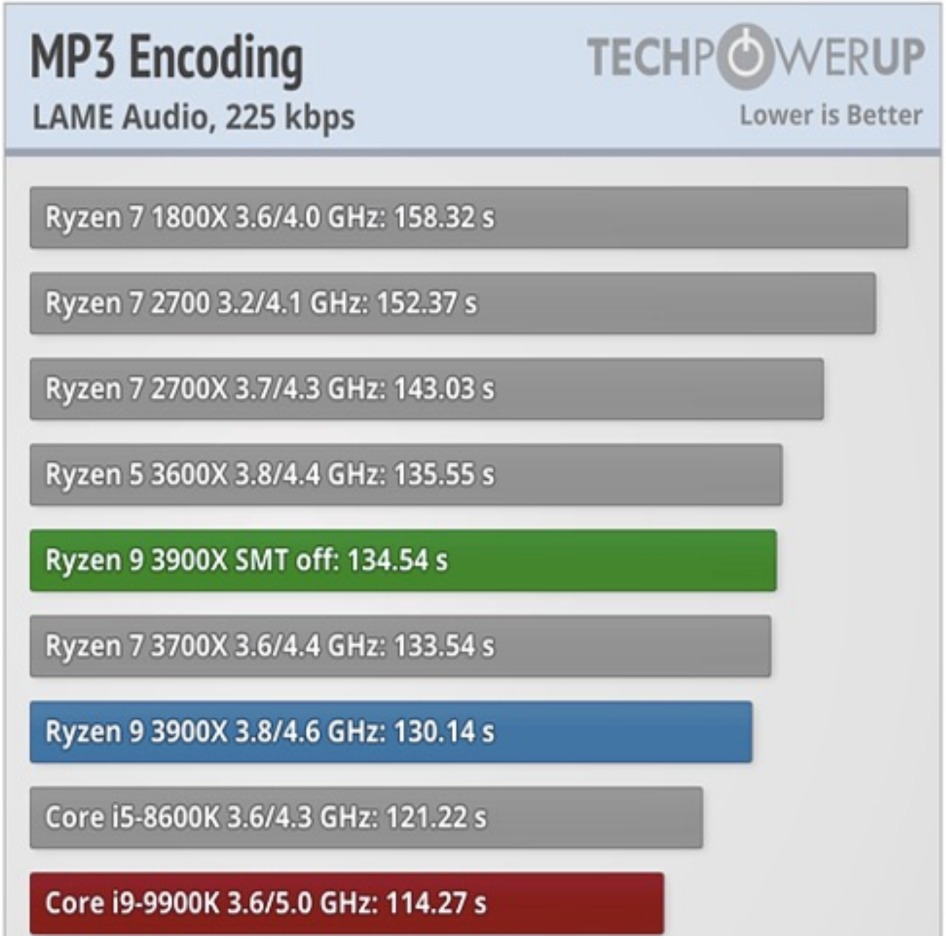
SIMD

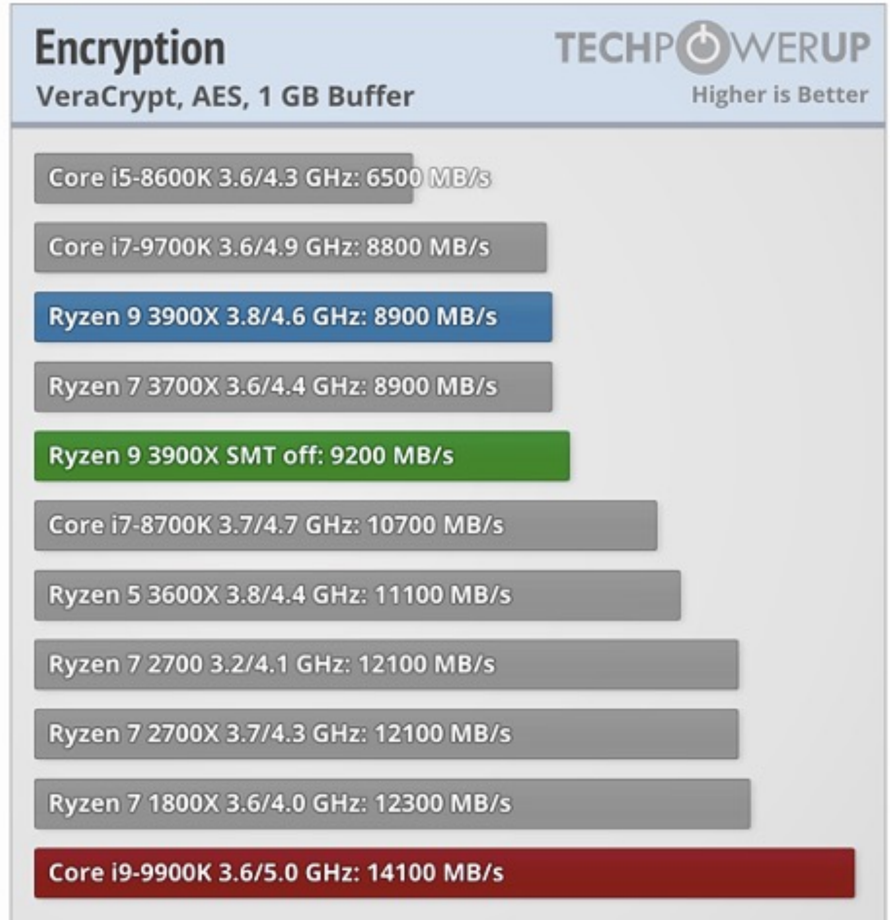

Normalized SIMD vectorization speedup
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
1.00
10.39x
1.00
4.27x
1.00
10.52x
1.00
12.45x
1.00
2.25x
2D 5 × 5 convolution
Back projection
N-body
Radar (1D convolution)
Tree search
No vectorization
Vectorization

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP222

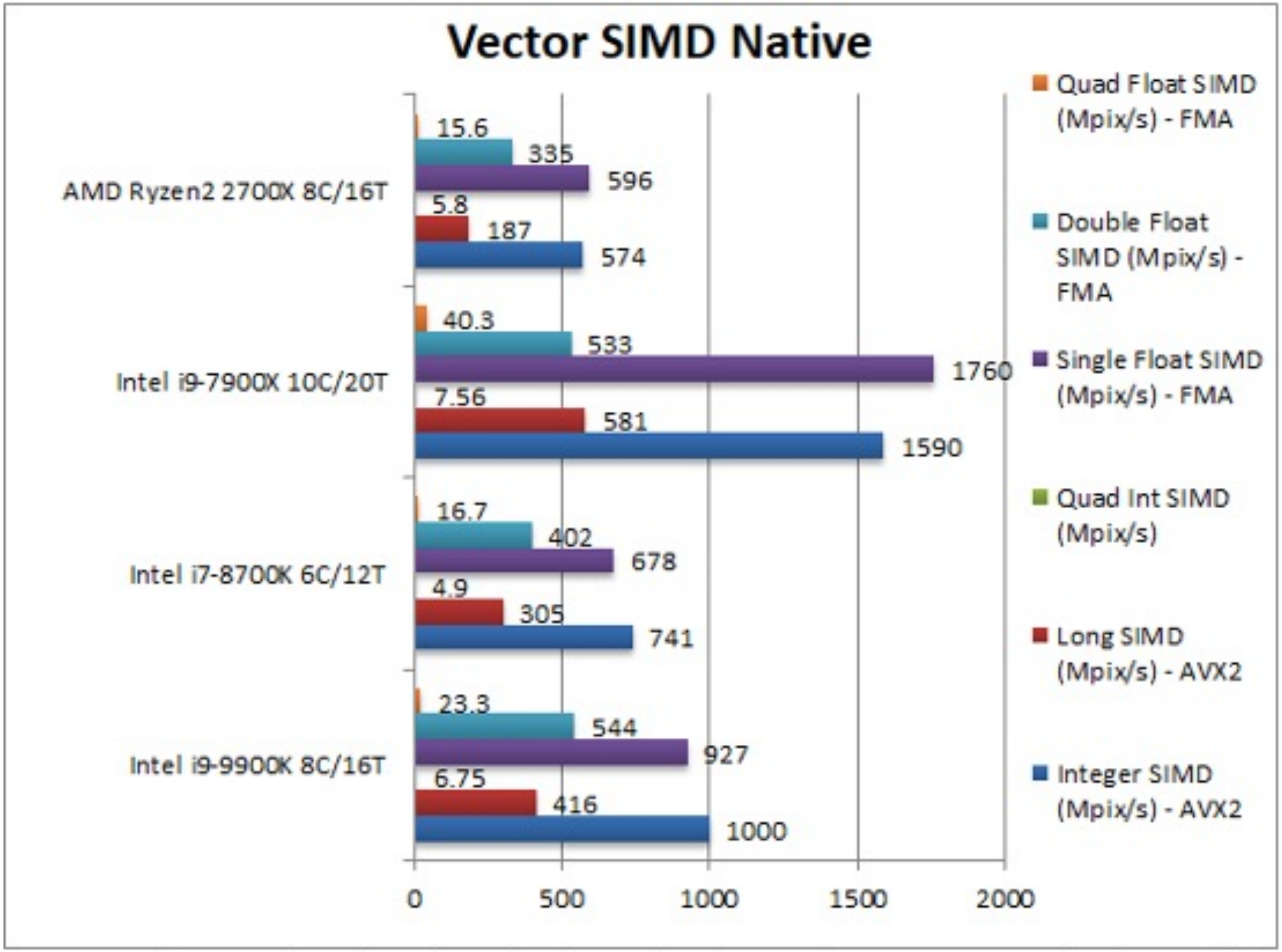
SIMD

DR JEFF SOFTWARE
INDIE APP DEVELOPER





SIMD



Here are the SIMD benchmarks. This shows that for Single Instruction Multiple Data loads, the i9-9900k beats AMD's 2700x. However, the bigger Skylake chip by Intel is still superior in SIMD loads.