CMOS Performance

Last updated 2/26/21

Performance

- What do we mean by Performance
  - Two primary performance parameters
  - Speed
    - the following discussion focuses on speed
  - Power
    - For mobile devices power is critical
      - want your laptop to last 5-8 hours
      - need your cell phone to last all day
      - want your mp3 player to last all week
      - want your e-reader to last a month
    - For Servers
      - cooling can be a significant expense for server farms
      - cooling is an issue for individual server "closets"

Performance

- Two primary speed measurements

  - Execution Time

    - How long it takes the processor to complete a task

    - Most familiar parameter to most of us
      - boot time
      - time to update the calculations on a large spreadsheet
      - time to read/write a file to disk
      - games – video updates and controller response time

    - In many cases the individual tasks are completed so fast we no longer perceive a delay
      - curser updates
      - directory traversal

Performance

- Two primary speed measurements
  - Throughput (bandwidth)
    - How many things can be completed in a fixed amount of time
    - Differentiated from execution time when –
      - Tasks can be performed in parallel
      - Portions of a task are dependent on outside resources
        - A processor that can jump to the next task while waiting on a read from disk will have higher throughput than one that must stall during the wait
        - Both take the same execution time to perform the task requiring the read but the first will also accomplish additional tasks during the same time.

Performance

- Two primary speed measurements

  - Improving (decreasing) execution time
    - Generally improves throughput
      - Each parallel or subtask completes faster
    - Exception: when there are not enough tasks to perform to fill the time
      - hurry-up and wait

  - Improving (increasing) throughput
    - Typically does not improve execution time
      - No one task completes any faster
    - Exception: when there are more tasks than can be completed in the allotted time queues will form
      - Assuming queue time is included in the execution time, improving throughput will improve execution time

Performance

- Theoretical CPI or IPC
  - CPI – Clocks per Instruction
    - Number of clocks required to execute a single instruction
    - Varies by instruction
    - Varies by ISA
  - HCS12: 1 to 4 clocks per instruction for most instructions
  - AVR: 1 clock per instruction for most instructions
  - Cortex A8: 0.5 clocks per instruction (dual issue)
  - Intel Core I7: 0.25 clocks per instruction (quad issue)
  - When the CPI gets below 1.0 – start to talk about IPC
    - Instructions per Clock
    - Cortex A8: IPC=2
    - Intel Core I7: IPC=4

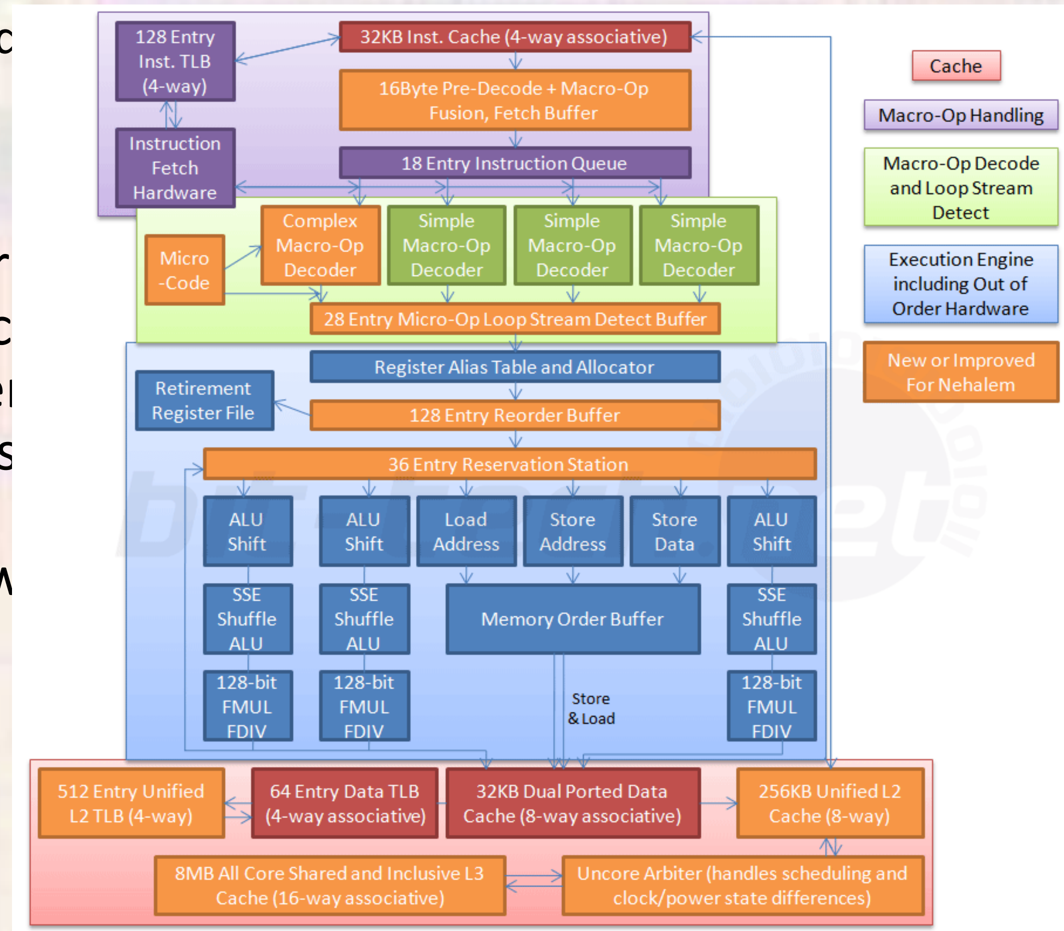
Performance

- Theoretical CPI or IPC

  - CPI – Clocks per Instruction
    - Number of clocks requirec
    - Varies by instruction
    - Varies by ISA

  - HCS12: 1 to 4 clocks per
  - AVR: 1 clock per instruc
  - Cortex A8: 0.5 clocks pe
  - Intel Core I7: 0.25 clocks

  - When the CPI gets below
    - Instructions per Clock
    - Cortex A8: IPC=2
    - Intel Core I7: IPC=4



Performance

- Practical CPI or IPC

  - CPI – Clocks per Instruction
    - Number of clocks in a program or program segment divided by the number of instructions executed

  - Varies from theoretical
    - Cache misses
    - Instruction distribution
    - Branch prediction errors

  - Impacted by
    - Architecture
    - Program
    - Compiler
    - Memory – hierarchy, size, speed

Performance

- Basic Calculations

$$CPU\ time\ (execuion\ time) = \frac{CPU\ clock\ cycles\ (for\ the\ task\ or\ program)}{Clock\ Rate}$$

Example 1: Task A requires 10,000 clock cycles on the ARM8 processor.
How long will this task take using an 800MHz clock?

$$CPU\ time\ = \frac{10{,}000\ clock\ cycles}{800 \times 10^6\ {}^{cycles}/_{s}} = 12.5\mu s$$

Example 2: Task B takes 2us when running on your 2GHz laptop.
You have the ability to modify the clock rate on your laptop.
What clock rate should you use to achieve a 1.5us execution time?

$$Clock\ Cycles = 2E9\frac{cycles}{s} \times 2\mu s = 4000\ clocks$$

$$Clock\ Rate = \frac{4000\ clock\ cycles}{1.5\mu s} = 2.666\ GHz$$

Performance

- Basic Calculations

$$CPU\ clock\ cycles = \text{Instructions for task} \times CPIaverage$$

Example: A program requires 10,000 instructions on the ARM8 processor.

Assuming a CPI_{ave} = 1.2, how many clock cycles will this program take?

$$CPU\ clock\ cycles = 10{,}000 \text{ instructions} \times 1.2\ {}^{clocks}/_{instruction} = 12K\ clocks$$

Performance

- Basic Calculations

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

Example: Program A requires 10,000 instructions using the ARM8 processor at 1.5GHz and a CPI =1.2.
How long will it take this program to run?

$$CPU\ time\ = \frac{10{,}000\ instructions \times 1.2\ {}^{clocks}/_{inst}}{1.5 \times 10^9\ {}^{clocks}/_{s}} = 8\mu s$$

Performance

- Basic Calculations

  - Amdahl's Law
    - The maximum expected improvement to an overall system when only part of the system is improved

$$CPU\ time\ after\ = \frac{Execution\ time\ affected\ by\ improvement}{Amount\ of\ improvement} + Execution\ time\ unaffected$$

Example:

Cache misses represent 20% of the overall execution time of your program.
You have developed a new cache that cuts the miss penalty in half.
How much will your program speed up?

$$CPU\ time\ new\ = \frac{0.2Ys}{2} + 0.8Ys = 0.9Ys$$ *, Y = current CPU time*

speed up = 10%

Performance

- Basic Calculations

  - MIPS
    - Measure of performance
    - Millions of Instructions / sec

    - Can only be used to compare processors of a common architecture
      - Different instructions → different # of instructions
      - Different CPIs → different clocks / instruction → different times

    - Can only be used to compare processors using the same program
      - Different programs on the same computer will lead to different MIPS measurements

Performance

- Benchmarks

  - Groups of programs designed to:
    - Exercise the various components of the processor
    - Emulate software representative of a more random application

  - Operate at the program level
    - Can be used across various processor architectures
    - Account for clock rates, memory compliments, accelerators

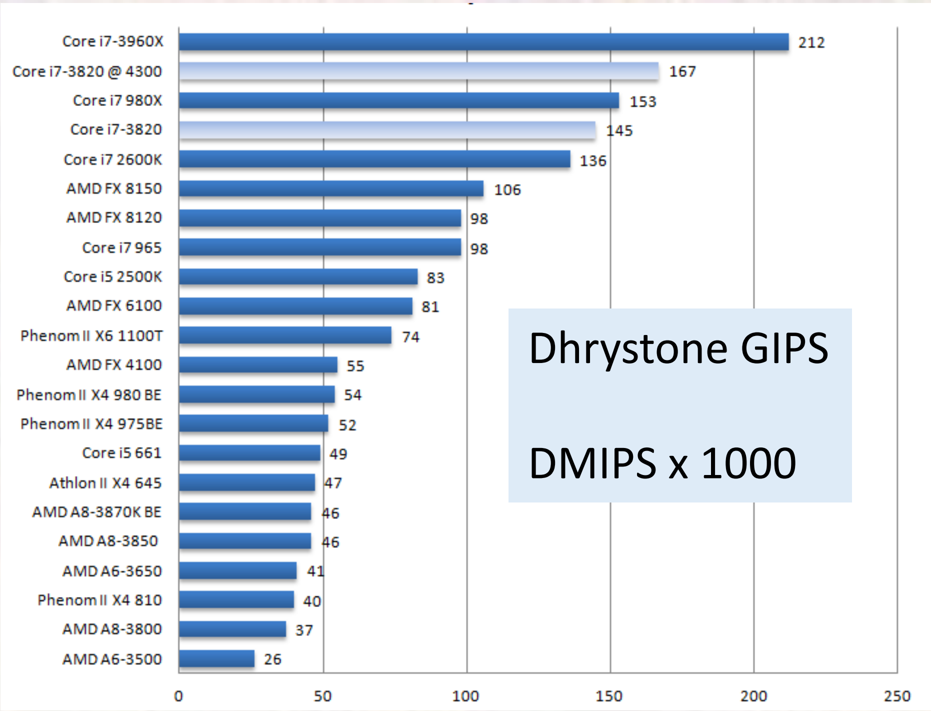
  - Can be manipulated
    - Compilers can target code to the benchmark, making a given implementation appear faster than it would normally be.

Performance

- Dhrystone
  - Dhrystone
    - Number of iterations of a loop of the benchmark code per second
  - Dhrystone VAX MIPs (Dhrystone MIPS or DMIPS)
    - Compare the performance of a processor against the performance of a reference machine
    - The benchmark is calculated by measuring the number of Dhrystones per second for the system, and then dividing that figure by the number of Dhrystones per second achieved by the reference machine (VAX11/780)
    - VAX 11/780 could execute 1750 Dhrystones/s
      - 1DMIP = 1750 Dhrystones/s
  - So “100 DMIPS" means “100 Dhrystone VAX MIPS", which means 100 times faster than a VAX 11/780
  - Measuring DMIPS/MHz removes clock frequency confusion

Performance

- Dhrystone
  - Limitations
    - No floating point operations
    - Easy to optimize compilers for the test



Notice – there is no clock frequency normalization here

Performance

- SPEC
  - Standard Performance Evaluation Corporation

  - SPEC CPU2017
    - Integer and Floating Point versions
    - SPECspeed – measure of time
      - 1 copy of a program
      - run time on reference system / time on system under test
      - bigger is better
    - SPECrate – measure of throughput
      - n copies of a program run in parallel
      - N* (run time on reference system / time on system under test)
      - bigger is better

Performance

- SPEC

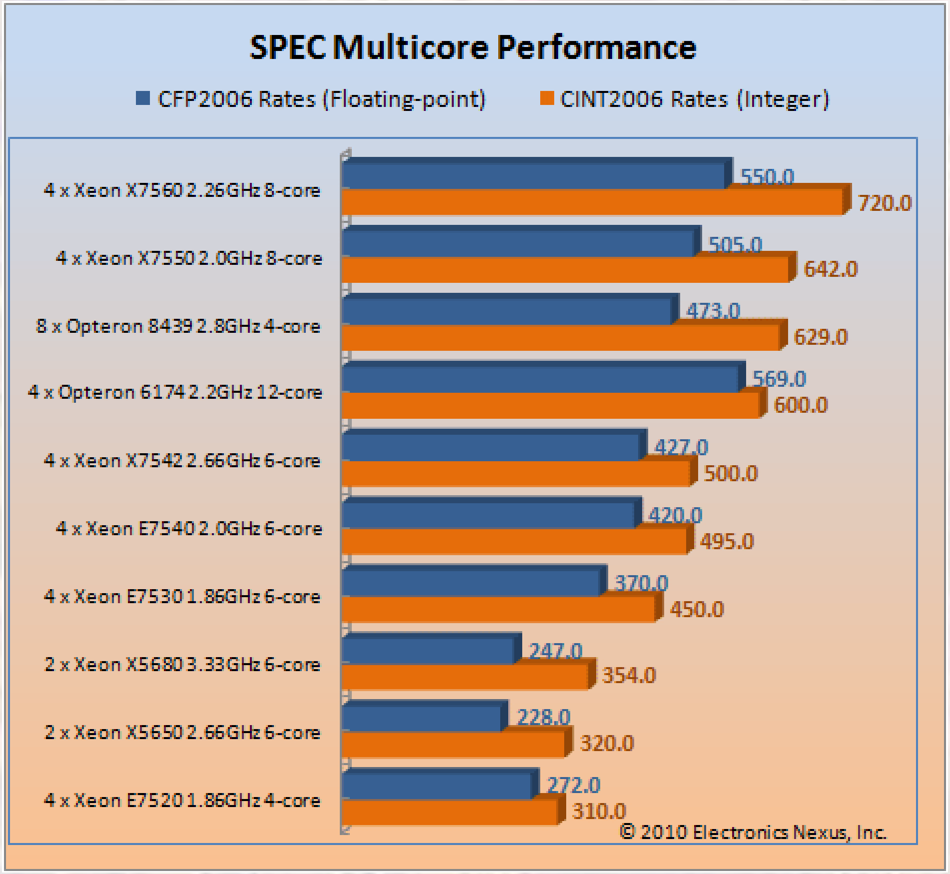
SPEC CPU 2017 has 43 benchmarks, organized into 4 suites:

SPECrate®2017 Integer	SPECspeed®2017 Integer	Language [1]	KLOC [2]	Application Area
500.perlbench_r	600.perlbench_s	C	362	Perl interpreter
502.gcc_r	602.gcc_s	C	1,304	GNU C compiler
505.mcf_r	605.mcf_s	C	3	Route planning
520.omnetpp_r	620.omnetpp_s	C++	134	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	96	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	33	General data compression

SPECrate®2017 Floating Point	SPECspeed®2017 Floating Point	Language [1]	KLOC [2]	Application Area
503.bwaves_r	603.bwaves_s	Fortran	1	Explosion modeling
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	257	Physics: relativity
508.namd_r		C++	8	Molecular dynamics
510.parest_r		C++	427	Biomedical imaging: optical tomography with finite elements
511.povray_r		C++, C	170	Ray tracing
519.lbm_r	619.lbm_s	C	1	Fluid dynamics
521.wrf_r	621.wrf_s	Fortran, C	991	Weather forecasting
526.blender_r		C++, C	1,577	3D rendering and animation
527.cam4_r	627.cam4_s	Fortran, C	407	Atmosphere modeling
	628.pop2_s	Fortran, C	338	Wide-scale ocean modeling (climate level)
538.imagick_r	638.imagick_s	C	259	Image manipulation
544.nab_r	644.nab_s	C	24	Molecular dynamics
549.fotonik3d_r	649.fotonik3d_s	Fortran	14	Computational Electromagnetics
554.roms_r	654.roms_s	Fortran	210	Regional ocean modeling

Performance

- SPEC
  - More common in larger processors



Performance

- COREMARK
  - Embedded Microprocessor Benchmark Consortium (EEMBC)
  - System focused benchmarks
    - Android, browser, TCP/IP
  - Processor focused benchmarks
    - COREMARK
    - variations for automotive, entertainment, low power, multiprocessors
  - “Coremark” is a measure of the number of iterations of the benchmark code loop per second

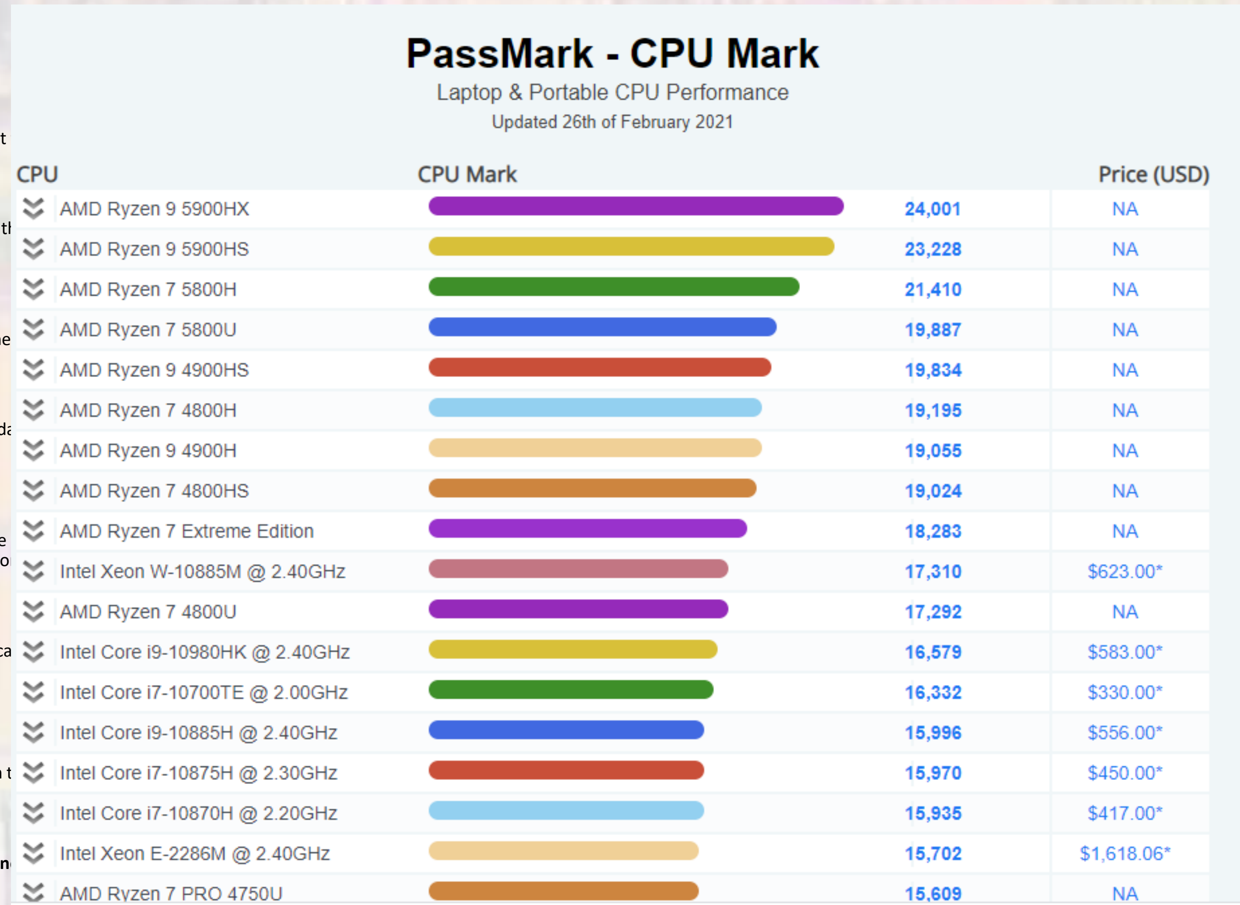
Performance

- COREMARK

Clear Sel.	Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	CoreMark / MHz	Threads	Date↓
☐	Intel Core i7-4750HQ		GCC4.2.1 Compatible Appl...	1600MHz DDR3L	2000	4	87886.33	43.94	8	2018-06-06
☐	Intel Core i7-4750HQ		GCC4.2.1 Compatible Appl...	1600MHz DDR3L	2000	4	17057.08	8.53	1	2018-06-06
☐	Intel I7-4700HQ		GCC4.8.2 20140120 (Red ...	32GB DDR3 799MHz	2400	1	17497.81	7.29	1	2017-04-25
☐	Intel I7-7700		GCC4.8.5 20150623 (Red ...	32GB DDR4 2133M...	3600	1	21775.42	6.05	1	2017-04-24
☐	Ineda I7		gcc4.9.1	LPDDR2 132Mhz	396	2	2882.00	7.28	4	2016-05-15
☐	Intel Core i7 860		Intel Parallel Studio XE 2015	DDR3-1333; Heap	2800	4	16622.34	5.94	1	2016-03-28
☐	Intel Core i7 860		GCC 4.9.2	8GB DDR3-RAM 13...	2800	4	55778.28	19.92	8	2015-02-19
☐	Intel i7-2640M		GCC4.5.3	DDR3	2800	1	14513.79	5.18	1	2013-04-03
☐	Intel i7-3612QE		Intel C++ 12.1	DDR3-1333	2100	4	83931.00	39.97	8	2013-01-30
☐	Intel Core i7-2760QM CPU @ 2.40GHz		GCC 4.5.3	DDR3 SDRAM 1333...	2400	4	85151.68	35.48	8	2012-10-19
☐	Intel Core i7-3930K CPU		GCC4.4.6 20110731 (Red ...	DDR3-1333 C9 32G...	3200	6	150962.39	47.18	12	2012-05-18
☐	Intel(R) Core i7-3930K CPU		GCC4.4.6 20110731 (Red ...	GCC4.4.6 20110731...	3200	1	116324.16	36.35	12	2012-05-18
☐	Intel Core i7 860		GCC4.5.1 20100924 (Red ...	DDR3 1333MHz	2801	1	51496.62	18.39	8	2011-09-05
☐	Intel(R) Core(TM) i7 CPU 870		GCC4.4.5	8GiB; DIMM 1333 ...	2930	1	51895.82	17.71	8	2011-03-24
☐	Intel Core i7 2600		GCC 4.4.5	DDR3-1333	3392.236	1	99562.34	29.35	16	2011-03-12
☐	Intel Core i7 720QM		gcc 4.4.5	DDR3 1066MHz; CL7	1600	1	31302.24	19.56	8	2011-02-17
☐	Intel Core i7 950		gcc (GCC) 4.1.2 20080704 ...	DDR-III/1066 CL9 H...	3060	1	48343.68	15.80	8	2011-02-16
☐	Intel Core i7 950		GCC3.4.4	PC3-10700 (667 M...	3600	1	57163.27	15.88	8	2010-05-13

Performance

- PassMark



- **Integer Math Test**
  - The **Integer Math Test** aims to measure how fast
- **Compression Test**
  - The **Compression Test** measures the speed that tl applications.
- **Prime Number Test**
  - The **Prime Number Test** aims to test how fast the
- **Encryption Test**
  - The **Encryption Test** encrypts blocks of random da power of'.
- **Floating Point Math Test**
  - The **Floating Point Math Test** performs the same to Integer numbers as well as being quite commo
- **Multimedia Instructions**
  - The **Multimedia Instructions** measures the **SSE** ca
  - SSE stands for Streaming SIMD extensions.
- **String Sorting Test**
  - The **String Sorting Test** uses the **qSort algorithm** t
- **Physics Test**
  - The **Physics Test** uses the **Tokamak Physics Engin**
- **Single Core Test**
  - The single core test only uses one CPU core and rates the computers performance under these conditions…many applications still only use one core so this is an important metric,

Performance

- Caveats

  - Most benchmarks measure a combination of CPU/system and compiler performance

  - Significant results variation depending on cache size
    - If the benchmark fits in the cache → better results

  - All benchmarks simulate a fixed amount of code and situations
    - Most processors are subject to wide variations in code