Raspberry Pi 3B+ 32 Bit and 64 Bit Benchmarks and Stress Tests

Roy Longbottom

Contents

Summary

Previously, I have run my 32 bit and 64 bit benchmarks on the appropriate range of Raspberry Pi computers, up to model 3B, and Operating Systems. Details of the benchmarks, results and download links are available from ResearchGate in this PDF file.

This report contains brief reminders of the benchmarks, with 32 bit and 64 bit results on a Rspberry Pi 3B+. On obtaining the computer, the original Operating Systems failed to boot. Raspbian required an update, but a new version of a 64 bit system was required. In this case, a working Gentoo became available.

Existing benchmarks were used, to provide comparisons between the old 3B model and the new 3B+ and 64 bit versus 32 bit operation. The new version of Gentoo was used on both hardware platforms to provide compatible results. The benchmarks and results are summarised as follows.

Single Core CPU Tests - comprising Whetstone, Dhrystone, Linpack and Livermore Loops Classic Benchmarks. Performance improvements of the 3B+, compared with the model 3B, were effectively proportional to increased CPU MHz, as expected. Possibly compiler version dependent 64/32 bit average gains were between 1.32 and 2.49 times.

Memory Benchmarks - measuring Floating point and integer performance using data from caches and RAM. Expected 3B+ gains were generally demonstrated with cached data, but could be slightly slower from RAM. In the numerous measurements, 32 bit compilations were sometimes faster than the 64 bit versions.

Multithreading Benchmarks - Most of the multithreading benchmarks execute the same calculations using 1, 2, 4 and 8 threads. These include some of the Classic Benchmarks, the most efficient was Whetstones, where each thread runs a nearly independent copy, and four could be completed in the same time as one. Next were Dhrystones, with some shared variables, providing 4 core gains of less than 2.5 times, then Linpack, demonstrating that the original code was completely unsuitable for multithreading, due to frequent thread starts and stops.

MP cache/RAM benchmarks are included, with serial and random access and sharing data, but starting at different points. Here serial reading provided adequate MP gains from cached data and some from RAM, then no gain with reading and writing and slow RAM random access. Then there is another, compiled using OpenMP, that providied totally confusing results.

Finally, a number of benchmarks attempt to measure maximum MFLOPS floating point speed, using the same series of calculations, with variants covering single and double precision (SP and DP), vector intrinsic functions and OpenMP. Performance is shown to radically improve with new compiler versions. Best 64 bit four core scores were 11.56 GFLOPS SP and 4.49 DP. Assembly code produced by the compilers is included later.

Java and OpenGL Benchmarks - A Java Whetstone benchmark is provided and one using JavaDraw functions. The latter has six tests with increasing demands, where the 3B+ was slightly faster than the 3B and 32 bit speeds faster than 64 bits, performance being dependent on Java and driver versions. The OpenGL benchmark has parameters to run at different window sizes, drawing coloured and textured objects, including an effective real design application. Results show similar performance using 3B and 3B+ running at both 32 and 64 bits, for the simpler graphics tests, with the 3B+ becoming faster with increasing complexity. The OpenGL program can also be used as a stress test.

Drive LAN and WiFi Benchmarks - Variations of the same progam are provided to benchmark internal and USB drives or LAN and WiFi connections, measuring performance using large files, small files and random access. The 64 bit drive version failed to run as expected (a direct I/O caching issue). Both 32 bit and 64 bit LAN/WiFi versions ran successfully on 3B and 3B+, communicating with Windows 7, Windows 10 and Linux Ubuntu. However, there were numerous complications and peculiar results that warrant further investigation. At least, LAN and WiFi large file data transfer speeds could be significantly faster via the 3B+.

Stress Test Summary below or Go To Start

Stress Tests - There are three main stress testing programs, that have parameters to select specific tests and running time, including one using integer calculations and another floating point, each with options to select cache or RAM sized data. The third is the OpenGL benchmark. To assist in resolving an earlier issue, I modified my CPU MHz/temperature monitor to include core voltage. Most sessions were run for 15 minutes.

The first tests used four copies of the processor programs, at both 32 and 64 bits, on the Pi 3B+ in a plastic case, with no CPU heatsink. These demonstrated the thermal characteristics, with the CPU MHz reducing from 1400 to 1200 (within a minute) at 70°C, with core voltage falling the same time. Later, thermal throttling kicked in on reaching 80°C, with MHz down to below 1000. Surprisingly, the floating point test performance reductions were less than those from the integer tests.

Running the OpenGL stress test (on a hot day) lead to performance reductions on reaching 70°C, after four minutes, but not much further. It was noted that more than one CPU core was being used and this affected later tests. The OpenGL test was one of those that ran slowly when an extended power cable was used, with reductions in MHz to 600, voltage to 1.2 and Frames Per Second from 20 to 8 (at a low 52.6°C).

The next tests comprised three CPU tests plus OpenGL, on the 3B+ at 64 bits. Using both the integer and floating point programs, temperatures exceeded 80°C for at least 10 minutes out of 15, worst OpenGL performance being with the latter program, again reducing to 8 FPS (at 1034 MHz, and 82.2°C).

FLIRC Case - The CPU and OpenGL tests were repeated with The Pi 3B+ board in an aluminium FLIRC case, that acts as an efficient heat sink. During two consecutive 15 minute tests, 1200 MHz was recorded occasionally, minimum FPS of 17 and CPU performance near maximum possible.

Assembly Code - In order to identify reasons for significant performance differences, disassembled code was produced, with that for critical processing loops provided. Main differences were between the use of scalar and vector instructions.

System ID - Details of the system used are identified using original standard functions. These show differences between 64 bit and 32 bit OS operation and Linux versions, but exactly the same for Pi 3B and 3B+ processors.

Whetstone Benchmark below or Go To Start

Whetstone Benchmark - whetstonePiA7, whetstonePi64

The Whetstone Benchmark was the first general purpose benchmark that set industry standards of performance, particularly for minicomputers, and introduced in 1972. The benchmark produced speed ratings in terms of Thousands of Whetstone Instructions Per Second (KWIPS). In 1978, self timing versions (by yours truly) produced speed ratings, for each of the eight test procedures, in MOPS (Millions of Operations Per Second) or MFLOPS (Millions of Floating Point Operations Per Second), with an overall rating in MWIPS, mainly dependent on floating point speed, perticularly the COS and EXP tests on the latest systems.

See Whetstone Benchmark History and Results and Whetstone Benchmark Detailed Later Results.

As for most benchmarks that completely depend on CPU speed, measured performance comparisons, at a given bit density, are essentially proportional to clock MHz, in this case 1400/1200. Performance at 64 bits is effectively the same as the 32 bit results, except for faster speeds in the tests that use such as COS and EXP functions. In turn, this leads to a superior 64 bit MWIPS rating.

As noted with my Android benchmarks, performance using Java can vary considerably with the version of run time software included. For comparable 3B and 3B+ speeds at 64 bits, running both under the later Gentoo upgrade was required. For the same reason, it is not appropriate to compare Java results between 32 bit and 64 bit systems.

System	MHz	MWIPS	MFLOPS			MOPS				
			1	2	3	COS	EXP	FIXPT	IF	EQUAL
32 Bit										
RPi 3 v8-A53	1200	711.6	336.5	329.7	256.9	12.2	8.8	1498.5	1796.7	1198.7
RPi 3B+ v8-A53	1400	829.9	392.7	384.6	299.8	14.2	10.2	1748.1	2095.8	1398.5
Ratio	1.17	1.17	1.17	1.17	1.17	1.16	1.16	1.17	1.17	1.17
Java										
Both Java 1,8.0_65										
RPi 3 v8-A53	1200	183.4	184.1	179.6	91.1	5.94	1.19	460.5	88.6	276.6
RPi 3B+ v8-A53	1400	211.8	214.2	207.6	105.8	6.92	1.37	535.5	103.1	321.3
Ratio	1.17	1.15	1.16	1.16	1.16	1.16	1.15	1.16	1.16	1.16
64 Bit										
RPi 3 v8-A53	1200	969.9	330.1	346.7	282.8	19.5	11.2	1459.9	######	1171.9
RPi 3B+ v8-A53	1400	1124.9	383.2	402.7	327.8	22.6	13.0	1699.5	######	1358.0
Ratio	1.17	1.16	1.16	1.16	1.16	1.16	1.16	1.16		1.16
3B+ 64/32 bit		1.36	0.98	1.05	1.09	1.59	1.27	0.98		0.97
Java										
Java 1.8.0_121, Linux 4.10.0										
RPi 3 v8-A53	1200	783.0	335.4	296.3	207.0	19.0	18.1	667.1	160.8	88.3
Both Java 1.8.0_161, Linux 4.14.31										
RPi 3 v8-A53	1200	667.9	268.9	249.6	112.2	20.0	18.8	608.8	207.8	76.7
RPi 3B+ v8-A53	1400	774.6	311.5	289.9	130.2	23.2	21.8	706.6	241.0	89.1
Ratio	1.17	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16

compiler optimiser produces 1 pass, this test does not affect MWIPS much

Dhrystone Benchmark below or Go To Start

Dhrystone 2 Benchmark - dhrystonePiA7, dhrystonePi64

The Dhrystone "C" benchmark provides a measure of integer performance (no floating point instructions). It became the key standard benchmark from 1984, with the growth of Unix systems. The first version was produced by Reinhold P. Weicker in ADA and translated to "C" by Rick Richardson. Two versions are available - Dhrystone versions 1.1 and 2.1. The second version, used here, was produced to avoid over-optimisation problems encountered with version 1, but some is still possible. Speed was originally measured in Dhrystones per second. This was later changed to VAX MIPS by dividing Dhrystones per second by 1757, the DEC VAX 11/780 result, the latter being regarded as the first 1 MIPS minicomputer.

See Dhrystone Results On PCs and Later Devices.

Here, 3B+/3B speeds are again proportional to CPU MHz. 64 bit speeds are indicated as being 40% faster than 32 bit, maybe due to more efficient instructions or benefiting from the availability of additional registers. Then the latter might allow more over-optimisation.

System		MHz	VAX MIPS	MIPS/MHz
32 Bit				
RPi 3	v8-A53	1200	2469	2.06
RPi 3B+	v8-A53	1400	2881	2.06
Ratio		1.17	1.17	
64 Bit				
RPi 3	v8-A53	1200	3475	2.90
RPi 3B+	v8-A53	1400	4021	2.87
Ratio		1.17	1.16	
3B+ 64/32 bit			1.40	

Linpack Benchmark below or Go To Start

Linpack Benchmark - linpackPiA7, linpackPiA7SP, linpackPi64, linpackPiSP64
Plus Linpack NEON - linpackPiNEONi, linpackPiNEONi64

The Linpack Benchmark was produced from the "LINPACK" package of linear algebra routines. It became the primary benchmark for scientific applications, particularly under Unix, from the mid 1980's, with a slant towards supercomputer performance. The original double precision C version, used here, operates on 100x100 matrices. Performance is governed by an inner loop in function daxpy() with a linked triad dy[i] = dy[i] + da * dx[i], and is measured in Millions of Floating Point Operations Per Second (MFLOPS).

Programming procedures and displayed output are the same as the original version for PCs (My accepted conversion at Netlib - 1996), where the bloated detail was needed due to using a low resolution timer. As for the original Fortran version, two sets of results are produced, with different memory alignment, and the lowest MFLOPS selected as the speed rating. This can lead to variation over multiple runs.

The benchmark produces a set of numeric results of calculations that demonstrate accuracy and consistency. These can vary, mainly by not much, as a result of the compiler generating different scalar or vector instructions. The source code includes the option of changing values used for comparison purposes, to suit particular situations. In this case, some benchmark programs have not been modified and result in an error message (see below). The range of results encountered is also shown.

See also Linpack Benchmark Results On PCs and Later Devices.

Besides compiled from standard C code, a new version is included, using NEON Intrinsic Functions for the daxpy function. This produces a significant performance gain with 32 bit compilation, but the vector instructions, used at 64 bits, provide similar speed gains.

Note the 64 bit performance gains in the table, that are up to near 2.5 times. Model 3B+/3B performance ratios are again mainly proportional to those for CPU MHz. MFLOPS per MHz ratios are also shown, now better than the Whetstone benchmark at up to 0.43 single precision and 0.29 double precision, for 64 bit programs.

		------ MFLOPS ----			--- MFLOPS/MHz --		
System	**MHz**	**DP**	**SP**	**NEON SP**	**DP**	**SP**	**NEON SP**
32 Bit							
RPi 3 v8-A53	1200	180	194	486	0.15	0.16	0.41
RPi 3B+ v8-A53	1400	210	226	562	0.15	0.16	0.40
Ratio	1.17	1.17	1.16	1.16			
64 Bit							
RPi 3 v8-A53	1200	343	484	521	0.29	0.40	0.43
RPi 3B+ v8-A53	1400	397	563	605	0.29	0.40	0.43
Ratio	1.17	1.20	1.17	1.17			
3B+ 64/32 bit		1.89	2.49	1.08			

Error Message Example

```
Variable norm. resid Non-standard result was              1.9 instead of              1.7
Variable resid       Non-standard result was   8.46778499e-14 instead of   7.41628980e-14
Variable x[0]-1      Non-standard result was  -1.11799459e-13 instead of  -1.49880108e-14
Variable x[n-1]-1    Non-standard result was  -9.60342916e-14 instead of  -1.89848137e-14
```

Results of Calculations

	norm resid	**resid**	**x[0]-1**	**x[n-1]-1**
DP Pi	1.7	7.41628980e-14	-1.49880108e-14	-1.89848137e-14
DP Pi 2-3	1.9	8.46778499E-14	-1.11799459E-13	-9.60342916E-14
DP Pi 64	1.9	8.46778499e-14	-1.11799459e-13	-9.60342916e-14
SP Pi	1.6	3.80277634e-05	-1.38282776e-05	-7.51018524e-06
SP Pi NEON	2.2	5.16722466e-05	-2.38418579e-07	-5.06639481e-06
SP Pi 2-3	2.0	4.69621336E-05	-1.31130219E-05	-1.30534172E-05
SP Pi 64	2.0	4.69621336e-05	-1.31130219e-05	-1.30534172e-05

Livermore Loops Benchmark below or Go To Start

Livermore Loops Benchmark - liverloopsPiA7, liverloopsPi64

This original main benchmark for supercomputers was first introduced in 1970, initially comprising 14 kernels of numerical application, written in Fortran. This was increased to 24 kernels in the 1980s. Performance measurements are in terms of Millions of Floating Point Operations Per Second or MFLOPS. The kernels are executed three times with different double precision data array sizes. Following are overall MFLOPS results for various systems, **geometric mean being the official average performance**.

The speed of the original Raspberry Pi could be rated as 4.5 times faster than the Cray 1 supercomputer - see my quote on Cost and Physical Differences. Now, one core of the Raspberry Pi 3B+ produces performance equivalent to 24 Cray 1 computers.

Some of the program's 3 x 24 kernels included produce inconsistent speeds, particularly for the minimum value but CPU MHz ratios still broadly apply to the performance summaries. The 64 bit official average MFLOPS rating is shown as being 32% faster than at 32 bits, with double precision MFLOPS/MHz at 0.20. The latter for maximum speed is 0.51.

See also Livermore Loops Benchmark Results On PCs and Later Devices.

32 Bit SUmmary

		DP MFLOPS					Per MHz
System	**MHz**	**Maximum**	**Average**	**Geomean**	**Harmean**	**Minimum**	**Geomean**
RPi 3 v8-A53	1200	398.4	210.6	185.9	160.2	56.5	0.15
RPi 3B+ v8-A53	1400	462.5	243.8	215.2	185.7	65.6	0.15
Ratio	1.17	1.16	1.16	1.16	1.16	1.16	

64 Bit Summary

RPi 3 v8-A53	1200	633.1	275.8	245.2	215.5	81.3	0.20
RPi 3B+ v8-A53	1400	720.6	320.2	285.6	251.9	94.4	0.20
Ratio	1.17	1.18	1.16	1.15	1.14	1.04	
3B+ 64/32 bit		1.59	1.31	1.32	1.35	1.44	

32 Bit DP MFLOPS 24 Loops

Raspberry Pi 3 1200 MHz

192.9	228.0	398.4	337.4	124.6	167.5	359.7	384.3	347.7	171.6	132.5	74.7
83.9	109.1	225.4	221.2	307.9	288.6	202.2	211.9	114.7	56.9	300.2	170.1

Raspberry Pi 3B+ 1400 MHz

223.8	264.4	462.5	392.9	146.0	159.4	416.0	446.3	406.7	199.1	153.8	86.7
99.5	126.7	261.8	256.8	357.4	333.6	234.7	239.5	132.9	66.0	345.4	197.5

Ratios 0.95 to 1.19, average 1.15 (Normal variations)

64 Bit DP MFLOPS 24 Loops

Raspberry Pi 3 1200 MHz

463.4	256.0	465.9	455.0	194.9	181.3	633.1	410.3	417.9	196.2	146.2	211.4
104.5	139.5	250.8	222.1	379.5	447.1	286.4	238.0	239.3	82.0	312.6	179.9

Raspberry Pi 3B+ 1400 MHz

538.9	297.5	539.8	528.6	225.5	208.6	720.6	477.9	475.9	252.1	169.7	245.2
127.2	159.7	290.9	258.2	441.1	509.4	332.9	279.9	302.9	95.5	337.4	208.9

Ratios 1.03 to 1.31, sverage 1.16 (Normal variations)
3B+ 64/32 bit 1.00 to 2.83, avsrage 1.40

Memory Speed Benchmark below or Go To Start

Memory Speed Benchmark - memspeedPiA7, memSpdPi64, memSpdPiNEON

MemSpeed benchmark measures data reading speeds in MegaBytes per second carrying out calculations on arrays of cache and RAM data, normally sized 2 x 4 KB to 2 x 4 MB. Calculations are as shown in the result headings. For the first two double precision tests, speed in Million Floating Point Operations Per Second (MFLOPS) can be calculated by dividing MB/second by 8 and 16. For single precision divide by 4 and 8. There is also a version that instructs the compiler to use NEON code. The 32 bit version results are provided below, but the particular compile options used were not acceptable using a 64 bit compiler.

In this case, relative 3B/3B+ speed ratios were calculated as separate averages for tests that use L1 cache, L2 cache and RAM. The cache based measurements were, as usual, equivalent to those derived from CPU MHz, but indicate that RAM could be slightly slower.

As would be expected, the use of NEON instructions provided a performance gain, using single precision floating point (32 bit system only). The 64/32 bit ratios are provided below, for the normal MemSpeed benchmarks, indicating the highest 64 bit gains were for double precision calculations, then single precision MFLOPS/MHz ratios, were similar to that derived from the NEON benchmark, 64 bit floating point calculations benefiting from using vector instructions. See details of Assembly Code.

The first two calculations are essentially the same as those in the Linpack benchmark performance dependent daxpy function, but speed not deflated by frequent calls to a function. This increases 64 bit MFLOPS/MHz to 0.43 double precision and 0.52 single precision.

############################ RPi 3 32 Bit ################################

Memory Reading Speed Test vfpv4 32 Bit Version 1

Memory KBytes Used	x[m]=x[m]+s*y[m] Dble MB/S	Sngl MB/S	Int+ Int32 MB/S	x[m]=x[m]+y[m] Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=y[m] Dble MB/S	Sngl MB/S	Int32 MB/S
	Raspberry Pi 3 CPU 1200 MHz, SDRAM 900 MHz								
8	1619	1812	3448	2375	2237	3793	2698	3121	3147
16	1621	1814	3459	2379	2240	3793	2710	3136	3162
32	1577	1743	3243	2277	2132	3138	2702	3123	3131
64	1537	1690	3126	2196	2047	3362	2566	2890	2917
128	1570	1714	3257	2243	2076	3502	2624	2993	3027
256	1573	1720	3285	2261	2084	3522	2652	3071	2930
512	1453	1598	2785	2055	1906	2081	2430	2783	2815
1024	918	1097	1327	1204	1185	1355	1606	1261	1263
2048	891	1032	1224	1133	1113	1191	882	811	817
4096	885	1023	1223	1127	1104	1201	787	756	755
8192	876	1019	1225	1118	954	1203	876	871	873
Max MFLOPS	203	454							
Per MHz	0.17	0.38							

############################ RPi 3B+ 32 Bit ################################

	Raspberry Pi 3B+ CPU 1400 MHz, SDRAM ?									**Avg. Gain**
8	1899	2125	4041	2783	2624	4448	3164	3693	3693	1.17 L1
16	1901	2128	4058	2791	2628	4462	3177	3703	3707	
32	1852	2049	3817	2686	2508	4161	3186	3715	3711	
64	1796	1959	3574	2542	2367	3855	2945	3347	3347	1.16 L2
128	1826	1989	3741	2600	2408	4031	3042	3506	3508	
256	1833	1995	3771	2617	2414	4068	2860	3616	3617	
512	1517	1618	2587	2039	1911	2687	2459	2825	2832	
1024	968	1098	1221	1172	1140	1211	1455	1144	1137	0.98 RAM
2048	911	980	1060	1038	1026	1062	1013	941	935	
4096	913	993	1064	1047	1038	948	992	902	903	
8192	926	1013	1077	1074	1065	1085	782	784	783	
Max MFLOPS	238	532								
Per MHz	0.17	0.38								

More Below

########################## RPi 3 NEON 32 Bit ##############################

Memory Reading Speed Test NEON 32 Bit Version 1 by Roy Longbottom

Memory KBytes Used	x[m]=x[m]+s*y[m] Int+ Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=x[m]+y[m] Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=y[m] Dble MB/S	Sngl MB/S	Int32 MB/S
	Raspberry Pi 3 CPU 1200 MHz, SDRAM 900 MHz								
8	1627	2387	3467	2387	3181	3812	2713	3164	3149
16	1621	2377	3457	2377	3169	3805	2713	3164	3165
32	1577	2273	3238	2280	2985	3535	2647	3103	3105
64	1526	2150	3018	2157	2793	3256	2568	2921	2921
128	1554	2217	3190	2216	2925	3436	2631	3028	3029
256	1561	2228	3225	2221	2948	3471	2654	3077	3077
512	1434	2010	2742	1978	2534	2313	2468	2840	2840
1024	950	1227	1324	1182	1306	1339	1581	1298	1298
2048	935	1136	1215	1128	1212	1214	915	880	885
4096	913	1121	1180	1131	1213	1212	825	844	842
8192	926	1134	1212	1126	936	1199	792	774	790
Max MFLOPS	203	594							
Per MHz	0.17	0.50							

########################## RPi 3B+ NEON 32 Bit ############################

										Avg. Gain
	Raspberry Pi 3B+ CPU 1400 MHz, SDRAM ?									
8	1890	2774	4027	2773	3694	4427	3130	3674	3676	1.16 L1
16	1885	2778	4037	2758	3702	4439	3155	3693	3691	
32	1813	2581	3646	2590	3366	3951	3130	3586	3591	
64	1808	2565	3653	2575	3370	3943	2987	3363	3366	1.17 L2
128	1790	2534	3606	2536	3334	3893	3040	3485	3485	
256	1796	2538	3638	2544	3360	3914	3079	3572	3569	
512	1654	2273	3163	2301	2945	3333	3010	3435	3447	
1024	959	1166	1185	1165	1209	1213	1438	1141	1130	0.97 RAM
2048	918	1059	1080	1061	1088	1081	1073	890	889	
4096	922	1076	1082	1069	1069	1084	1015	867	871	
8192	929	1089	1091	1083	1102	1081	786	774	774	
Max MFLOPS	236	695								
Per MHz	0.17	0.50								

########################## RPi 3 Gentoo 64 Bit ############################

Memory Reading Speed Test armv8 64 Bit

Raspberry Pi 3B CPU 1200 MHz, SDRAM 900 MHz

Memory KBytes Used	x[m]=x[m]+s*y[m] Int+ Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=x[m]+y[m] Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=y[m] Dble MB/S	Sngl MB/S	Int32 MB/S
8	4161	2506	3749	5347	3393	4166	4641	3730	3731
16	4032	2506	3758	5357	3419	4162	4674	3753	3750
32	4016	2486	3721	5311	3390	4137	4673	3714	3727
64	3372	2342	3361	4232	3123	3685	4244	3522	3499
128	3352	2393	3454	4266	3189	3789	4359	3564	3563
256	3227	2398	3463	4266	3224	3769	4246	3525	3525
512	633	2010	2885	3603	2674	2457	3733	3081	3084
1024	560	889	1217	1192	1202	1011	857	1094	1095
2048	565	880	1145	1131	991	1156	844	885	788
4096	514	1092	987	1127	1134	1159	873	944	951
8192	531	887	1150	1139	1038	1162	782	799	704
Max MFLOPS	520	627							
Per MHz	0.43	0.52							

More Below

######################## RPi 3B+ Gentoo 64 Bit ##########################

Raspberry Pi 3B+ CPU 1400 MHz, SDRAM ? **Avg. Gain**

8	4822	2888	4346	6190	3955	4830	5372	4324	4325	1.16 L1
16	4684	2904	4337	6197	3955	4833	5389	4340	4343	
32	4471	2898	4345	6172	3951	4824	5438	4323	4347	
64	3814	2630	3721	4671	3467	4052	5272	4238	4208	1.18 L2
128	3866	2727	3905	4797	3601	4257	4935	4102	4103	
256	3891	2765	3975	4877	3700	4296	4901	4096	4102	
512	671	2305	3252	3791	3003	3530	3638	3721	3718	
1024	694	1263	1324	1320	1317	1277	1192	1482	1477	1.18 RAM
2048	645	1213	1255	1245	1132	1269	840	921	925	
4096	617	1204	1122	1230	1238	1120	968	990	990	
8192	658	1210	1256	1224	1101	1271	1011	1082	1084	
Max MFLOPS	602	726								
Per MHz	0.43	0.52								

######################## RPi 3B+ Gentoo NEON 64 Bit ########################

Compile options to use NEON instructions are not available at 64 bit working.

#################### Compare 64 bit / 32 bit Pi 3 ######################

Memory Reading Speed Test

Memory KBytes Used	x[m]=x[m]+s*y[m] Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=x[m]+y[m] Dble MB/S	Sngl MB/S	Int32 MB/S	x[m]=y[m] Dble MB/S	Sngl MB/S	Int32 MB/S
8	2.57	1.38	1.09	2.25	1.52	1.10	1.72	1.20	1.19
256	2.05	1.39	1.05	1.89	1.55	1.07	1.60	1.15	1.20
8192	0.61	0.87	0.94	1.02	1.09	0.97	0.89	0.92	0.81

#################### Compare 64 bit / 32 bit Pi 3B+ ######################

8	2.54	1.36	1.08	2.22	1.51	1.09	1.70	1.17	1.17
256	2.12	1.39	1.05	1.86	1.53	1.06	1.71	1.13	1.13
8192	0.71	1.19	1.17	1.14	1.03	1.17	1.29	1.38	1.38

NEON Memory Speed Benchmark below or Go To Start

NEON Float & Integer Benchmark - NeonSpeed, NeonSpeedPi64

This was my first benchmark produced to measure speed using NEON instructions on ARM v7 CPUs using Android. It executes some of the code used in Memory Speed Benchmark, with additional tests recoded using NEON intrinsic functions. In this case there are no double precision calculations.

Pi 3B+ CPU/Cache performance gains are again proportional to MHz, with some worse via RAM. Single precision MFLOPS per MHz increased up to 0.92 through using NEON intrinsic functions. These were compiled as different vector instructions, including the use of the Fused Multiply Accumulate variety. See details of Assembly Code.

No significant 64 bit performance gains were indicated, using these test functions, as similar instructions were generated. Some of the 32 bit functions were also somehat faster.

###################### RPi 3 32 Bit ##########################

NEON Speed Test V 1.0

Raspberry Pi 3 CPU 1200 MHz

Vector Reading Speed in MBytes/Second

Memory KBytes	Float v=v+s*v Norm	Float v=v+s*v Neon	Int v=v+v+s Norm	Int v=v+v+s Neon	Neon v=v+v Float	Neon v=v+v Int
16	2720	4001	3459	4225	4474	4750
32	2598	3706	3268	3879	4091	4320
64	2453	3389	3069	3526	3675	3859
128	2503	3466	3178	3598	3718	3918
256	2530	3516	3230	3649	3779	3950
512	2221	2923	2718	2964	3104	3217
1024	1262	1326	1317	1316	1324	1316
4096	1170	1213	1204	1213	1210	1195
16384	1177	1229	1218	1147	1222	1215
65536	1181	1226	1221	916	1208	1218
Max MFLOPS	680	1000				
Per MHz	0.57	0.84				

###################### RPi 3B+ 32 Bit ########################

Raspberry Pi 3B+ CPU 1400 MHz

Memory KBytes	Float v=v+s*v Norm	Float v=v+s*v Neon	Int v=v+v+s Norm	Int v=v+v+s Neon	Neon v=v+v Float	Neon v=v+v Int	Avg Gain
16	3188	4690	4055	4953	5243	5570	1.17 L1
32	3143	4578	3990	4811	5120	5431	
64	2927	4089	3693	4253	4446	4674	1.16 L2
128	2864	3912	3588	4060	4172	4478	
256	2905	3953	3632	4119	4213	4524	
512	2255	2835	2661	2873	2922	3035	
1024	1234	1264	1263	1265	1248	1232	0.93 RAM
4096	1099	1114	1110	1106	1091	1088	
16384	1116	1128	1116	1117	1102	1092	
65536	1113	1132	1122	837	1107	1090	
Max MFLOPS	797	1173					
Per MHz	0.57	0.84					

################### RPi 3 Gentoo 64 Bit ####################

NEON Speed Test armv8 64 Bit V 1.0

Raspberry Pi 3 CPU 1200 MHz

Vector Reading Speed in MBytes/Second

Memory KBytes	Float v=v+s*v Norm	Float v=v+s*v Neon	Int v=v+v+s Norm	Int v=v+v+s Neon	Neon v=v+v Float	Neon v=v+v Int
16	2350	4419	3415	4176	4686	4843
32	2247	3991	3216	3831	4258	4348
64	2161	3631	3038	3559	3886	3882
128	2212	3744	3148	3648	3980	3966
256	2230	3766	3171	3677	4009	3962
512	1931	2736	2685	2663	3267	3322
1024	1116	1116	1223	1135	1156	1213
4096	1065	1075	1146	1040	1117	1162
16384	1065	1072	1149	978	1106	1078
65536	1007	1150	1076	824	1103	1137
Max MFLOPS	588	1105				
Per MHz	0.49	0.92				

More Below

################## RPi 3B+ Gentoo 64 Bit ###################

	Raspberry Pi 3B+ CPU 1400 MHz						Avg Gain
16	2724	5109	3961	4841	5446	5607	1.16 L1
32	2612	4645	3726	4450	4968	5036	
64	2523	4247	3540	4150	4521	4519	1.16 L2
128	2583	4363	3666	4253	4616	4635	
256	2576	4314	3674	4254	4591	4631	
512	1852	2871	2608	2466	2916	2698	
1024	1222	1207	1305	1179	1280	1216	1.08 RAM
4096	1157	1144	1214	1109	1181	1160	
16384	1175	1245	1244	1134	1191	1180	
65536	1143	1258	1185	909	1144	1260	
Max MFLOPS	681	1277					
Per MHz	0.49	0.91					

################ Compare 64 bit / 32 bit Pi 3 #################

Memory	Float v=v+s*v		Int v=v+v+s		Neon v=v+v	
KBytes	Norm	Neon	Norm	Neon	Float	Int
16	0.86	1.10	0.99	0.99	1.05	1.02
256	0.88	1.07	0.98	1.01	1.06	1.00
65536	0.85	0.94	0.88	0.90	0.91	0.93

############## Compare 64 bit / 32 bit Pi 3B+ ################

16	0.85	1.09	0.98	0.98	1.04	1.01
256	0.89	1.09	1.01	1.03	1.09	1.02
65536	1.03	1.11	1.06	1.09	1.03	1.16

Bus Speed Benchmark below or Go To Start

Bus Speed Benchmark - busspeedPiA7, busSpdPi64

This benchmark is designed to identify reading data in bursts over buses and possible maximum data transfer speed from RAM (using 1 core - see MP version). The program starts by reading a word (4 bytes) with an address increment of 32 words (128 bytes) before reading another word. The increment is reduced by half on successive tests, until all data is read.

Model 3B+ Speed gains are provided for reading all data, as usual similar to increase in MHz, with RAM speed ratio much less, but not negative. The 64 bit compiler produced unexpected slower speeds on reading all data from L1 cache, compared with addressing increment of 2 words. This leads to an indication that the 32 bit program is faster, using this test. As indicted for the MP-BusSpd Benchmark, the 64 bit compiler did not identify that vector SIMD instructions could be used.

###################### RPi 3 32 Bit ##########################

BusSpeed vfpv4 32b V1

Raspberry Pi 3 CPU 1200 MHz

Reading Speed 4 Byte Words in MBytes/Second

Memory KBytes	Inc32 Words	Inc16 Words	Inc8 Words	Inc4 Words	Inc2 Words	Read All
16	3335	3741	4075	4371	4388	4413
32	1964	2229	2787	4271	4308	4311
64	612	615	1121	1932	2880	3546
128	570	573	1034	1803	2756	3467
256	541	544	995	1758	2737	3457
512	382	408	794	1360	2269	3105
1024	128	136	256	533	1025	1945
4096	109	125	245	482	961	1585
16384	120	125	241	477	964	1744
65536	120	125	243	477	947	1881

###################### RPi 3B+ 32 Bit ########################

Raspberry Pi 3B+ CPU 1400 MHz

						Gain Read All
16	3751	4125	4755	4965	5083	5104	1.16 L1
32	1983	2177	2819	4258	4681	4958	
64	719	728	1333	2298	3428	4165	1.17 L2
128	664	666	1201	2130	3285	4084	
256	625	635	1163	2055	3197	4032	
512	329	360	702	1309	2297	3342	
1024	128	143	279	548	1061	2128	1.00 RAM
4096	115	131	256	498	978	1694	
16384	124	130	254	489	994	1620	
65536	126	129	253	492	1003	1728	

################### RPi 3 Gentoo 64 Bit ####################

Raspberry Pi 3 CPU 1200 MHz

BusSpeed armv8 64 Bit Mon

Reading Speed 4 Byte Words in MBytes/Second

Memory KBytes	Inc32 Words	Inc16 Words	Inc8 Words	Inc4 Words	Inc2 Words	Read All
16	3312	3684	4007	4341	4390	3341
32	2019	2158	2687	4172	4235	3294
64	577	595	1124	1861	2836	3062
128	546	556	1040	1754	2733	3062
256	516	530	1000	1696	2692	3094
512	341	272	708	1264	2099	2626
1024	77	126	251	488	847	1860
4096	85	115	222	446	908	1685
16384	99	115	231	393	902	1704
65536	98	115	229	443	810	1700

More Below

################## RPi 3B+ Gentoo 64 Bit ###################

	Raspberry Pi 3B+ CPU 1400 MHz						Gain Read All
16	3823	4251	4638	4945	5045	3854	1.15 L1
32	1543	1677	2423	3331	4152	3680	
64	672	694	1306	2169	3300	3577	1.17 L2
128	635	648	1211	2055	3202	3604	
256	600	615	1163	1971	3152	3612	
512	328	278	695	1272	2256	2978	
1024	94	140	281	543	960	2075	1.12 RAM
4096	99	128	259	448	1016	1931	
16384	125	129	258	500	898	1863	
65536	125	114	257	500	1015	1898	

################ Compare 64 bit / 32 bit Pi 3 ##################

Memory KBytes	Inc32 Words	Inc16 Words	Inc8 Words	Inc4 Words	Inc2 Words	Read All
16	0.99	0.98	0.98	0.99	1.00	0.76
256	0.95	0.97	1.01	0.96	0.98	0.89
65536	0.82	0.92	0.94	0.93	0.86	0.90

############## Compare 64 bit / 32 bit Pi 3B+ #################

16	1.02	1.03	0.98	1.00	0.99	0.76
256	0.96	0.97	1.00	0.96	0.99	0.90
65536	0.99	0.88	1.02	1.02	1.01	1.10

FFT Benchmarks below or Go To Start

FFT Benchmarks - fft1-RPi2, fft3c-Rpi2, fft1-RPi64, FFT3c-RPi64

There are two benchmarks, FFT1, the original, and FFT3c, optimised, with 32 bit and 64 bit versions, when appropriate. Performance is measured in milliseconds, for FFTs sized 1K to 1024K, with three measurements using both single and double precision floating point data, plus some sumchecks for the largest ones.

The second of the three measurements are provided below. Note that three of the smaller FFT tests can be executed in less than a millisecond, when the CPU MHz scaling governor can produce a lower frequency (64 bit system), leading to increased running time, until the high MHz kicks in (see example below). For full speed, the scaling governor setting should be performance (sudo su echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor).

Much of the data is accessed on a skipped sequential basis, where only part of data transferred in bursts, over buses, is likely to be used. The 3C version was optimised to use more of the burst data, producing much improved performance.

Data transfer covers caches and RAM. RPi 3B+ gains are provided for each FFT size, indicating where RAM transfers apply, when the performance ratio is less than that derived from CPU MHz. Comparisons of 64/32 bit performance are also shown, with some good and some bad. Note that the processing activity is unlikely to produce absolutely consistent speeds, particularly when data size is near cache capacity or execution time is very low.

################## FFT V 1 32 Bit ####################

	RPi3		RPi3B+		Compare	
Size	-------- milliseconds --------				3B+ Gain	
K	Single	Double	Single	Double	Single	Double
1	0.16	0.16	0.14	0.14	1.16	1.15
2	0.37	0.42	0.34	0.34	1.09	1.24
4	1.01	1.09	0.88	0.94	1.14	1.16
8	2.25	2.51	1.96	2.17	1.15	1.15
16	5.29	5.85	4.56	5.16	1.16	1.13
32	12.57	22.48	10.48	19.52	1.20	1.15
64	44.59	110.41	36.67	130.32	1.22	0.85
128	217.33	269.62	239.81	314.27	0.91	0.86
256	525.92	615.26	584.42	705.36	0.90	0.87
512	1199.23	1364.15	1324.23	1534.86	0.91	0.89
1024	2538.17	2831.33	2740.23	3152.95	0.93	0.90

################## FFT V 3C 32 Bit ####################

1	0.20	0.16	0.17	0.14	1.16	1.19
2	0.46	0.37	0.38	0.32	1.21	1.17
4	1.28	0.89	1.07	0.77	1.19	1.15
8	2.32	2.05	2.13	1.89	1.09	1.08
16	5.36	5.98	4.57	5.83	1.17	1.03
32	12.47	15.48	10.77	15.48	1.16	1.00
64	31.08	36.99	29.05	37.25	1.07	0.99
128	72.02	84.24	70.05	85.03	1.03	0.99
256	160.48	193.81	160.68	199.34	1.00	0.97
512	367.71	426.24	364.53	437.72	1.01	0.97
1024	799.23	948.54	794.54	974.48	1.01	0.97

RPi3B+ FFT3C scaling_governor ondemand

1	0.40	0.14
2	0.93	0.32
4	1.97	0.75
8	4.64	1.76
16	4.47	5.83

################## FFT V 1 64 Bit ####################

	RPi3		RPi3B+		Compare	
Size	-------- milliseconds --------				3B+ Gain	
K	Single	Double	Single	Double	Single	Double
1	0.18	0.18	0.15	0.15	1.18	1.17
2	0.35	0.39	0.29	0.41	1.21	0.96
4	0.87	1.64	0.79	0.99	1.10	1.65
8	2.08	3.18	1.87	2.45	1.12	1.30
16	4.68	7.18	3.86	5.23	1.21	1.37
32	10.76	29.77	10.20	23.64	1.05	1.26
64	39.65	126.03	50.50	105.28	0.79	1.20
128	174.53	302.94	148.80	262.45	1.17	1.15
256	408.05	700.83	352.60	603.27	1.16	1.16
512	956.18	1543.35	836.47	1362.20	1.14	1.13
1024	2055.48	3278.12	1841.71	2848.54	1.12	1.15

More Below

################## FFT V 3C 64 Bit ##################

1	0.18	0.19	0.14	0.18	1.30	1.05
2	0.41	0.43	0.36	0.37	1.13	1.17
4	0.80	0.99	0.70	0.86	1.15	1.15
8	2.10	2.32	2.60	1.95	0.81	1.19
16	6.22	5.66	4.66	5.05	1.33	1.12
32	10.38	15.08	9.05	13.20	1.15	1.14
64	27.59	35.71	24.54	31.38	1.12	1.14
128	71.14	81.19	56.37	72.85	1.26	1.11
256	139.33	190.07	124.65	170.73	1.12	1.11
512	321.96	428.64	295.34	385.23	1.09	1.11
1024	705.92	938.42	629.95	838.97	1.12	1.12

############## Compare 64 bit / 32 bit ################

	RPi3		RPi3B+	
K	**Single**	**Double**	**Single**	**Double**
FFT1				
1 to 8	1.05	0.86	1.06	0.90
16 to 128	1.17	0.83	1.14	1.06
256 to 1K	1.26	0.88	1.58	1.13
FFT3C				
1 to 8	1.24	0.89	1.17	0.88
16 to 128	1.05	1.04	1.15	1.17
256 to 1K	1.14	1.01	1.26	1.16

Next MultiThreading Benchmarks or Go To Start

MultiThreading Benchmarks

Most of the multithreading benchmarks execute the same calculations using 1, 2, 4 and 8 threads. One of them, MP-MFLOPS, is available in two different versions, using standard compiled "C" code for single and double precision arithmetic. A further version uses NEON intrinsic functions. Another variety uses OpenMP procedures for automatic parallelism.

As before, details of the benchmarks, other results and download links are available from ResearchGate in this PDF file.

On running my multithreading benchmarks, I noted unusual slow performance from certain tests. The first was the MP-Whetstone Benchmark, with independent copies of the program, using 1, 2, 4 and 8 threads. Then, the running time should not increase much using up to 4 threads, but should be just over twice as long using 8. As shown in the example below, the 4 thread test was too slow and this was particularly due to the long running COS test.

```
 MP-Whetstone Benchmark armv8 64 Bit Mon Jun 18 23:09:29 2018

                    Using 1, 2, 4 and 8 Threads

     MWIPS MFLOPS MFLOPS MFLOPS   Cos   Exp     Fixpt      If  Equal
                 1      2      3  MOPS  MOPS   ## MOPS    MOPS   MOPS

1T  1112.9  352.1  379.0  319.2  22.0  12.7 1641076.6  2722.5 1328.7
2T  2250.7  717.5  767.4  656.4  44.5  25.5 2684285.1  5456.3 2652.7
4T  2899.0 1342.3 1525.3 1048.1  42.6  46.1 1959513.0  4497.3 4319.1
8T  3433.1 1654.1 1804.6 1106.4  55.2  47.8 2453184.1 10960.3 4994.0

  Overall Seconds   5.14 1T,   5.11 2T,   8.08 4T,  13.66 8T

 ## over optimised but always had little effect on overall MWIPS
```

A (not official) 2.5 amp power supply was used and this was connected via a digital meter that measures current and voltage. During the tests, this reported constant over 5 volts and less than 1 amp. I suspected overheating and ran my RPiHeatMHz program at the same time, (See OpenGL Power Cable Tests for core volts) producing the results below and showing that the CPU MHz flipped into 600 MHz at the time of slow recorded performance. Although the temperature was not excessive. I carried out further tests with the system wrapped in bags of frozen food. The failures still occurred with recorded temperatures of less than 30°C.

```
Temperature and CPU MHz Measurement

Start at Mon Jun 18 23:09:26 2018

Using 40 samples at 1 second intervals

Seconds
   0.0     1400 scaling MHz,   1400 ARM MHz, temp=55.8°C
   1.0     1400 scaling MHz,   1400 ARM MHz, temp=55.8°C
   2.2     1400 scaling MHz,   1400 ARM MHz, temp=55.8°C
   3.3     1400 scaling MHz,   1400 ARM MHz, temp=56.4°C 1T
   4.5     1400 scaling MHz,   1400 ARM MHz, temp=56.9°C
   5.7     1400 scaling MHz,   1400 ARM MHz, temp=56.9°C
   6.9     1400 scaling MHz,   1400 ARM MHz, temp=56.9°C
   8.0     1400 scaling MHz,   1400 ARM MHz, temp=57.5°C 2T
   9.2     1400 scaling MHz,   1400 ARM MHz, temp=58.0°C
  10.4     1400 scaling MHz,   1400 ARM MHz, temp=58.0°C
  11.7     1400 scaling MHz,   1399 ARM MHz, temp=59.1°C
  12.9     1400 scaling MHz,   1400 ARM MHz, temp=59.1°C 4T
  14.1     1400 scaling MHz,    600 ARM MHz, temp=59.1°C
  15.4     1400 scaling MHz,    600 ARM MHz, temp=59.1°C
  16.8     1400 scaling MHz,    600 ARM MHz, temp=58.0°C
  18.3     1400 scaling MHz,   1400 ARM MHz, temp=60.1°C
  19.6     1400 scaling MHz,   1400 ARM MHz, temp=60.7°C
  20.8     1400 scaling MHz,   1400 ARM MHz, temp=61.2°C
  22.0     1400 scaling MHz,   1400 ARM MHz, temp=61.8°C 8T
  23.3     1400 scaling MHz,    600 ARM MHz, temp=60.1°C
  24.9     1400 scaling MHz,    600 ARM MHz, temp=60.1°C
  26.4     1400 scaling MHz,   1400 ARM MHz, temp=60.7°C
  27.6     1400 scaling MHz,   1400 ARM MHz, temp=61.2°C
  To
  38.8     1400 scaling MHz,   1400 ARM MHz, temp=60.1°C
```

Next, I tried using my official Pi 2 amp power supply and that seemed to be fine, but caused the failures when the meter was included, needing connection using a longer wire. It also failed when just the wire extension was included. Now all multithreading programs have been run to verify the results, using a directly connected official 2.5 amp power supply. Even with this, four thread performance can be inconsistent, mainly when the running time is not very long, influenced by other system activity and the programs calculating performance based on the last thread to finish. In these cases, four threads carry out the same number of instructions as a single thread, potentially reducing running time by a quarter. The Whetstone benchmark is probably the best one to identify the power drop, with each thread (of up to four) taking around 5 seconds to execute the same functions.

MP-MFLOPS Benchmarks
MP-MFLOPSPiA7, MP-MFLOPSDP, MP-MFLOPSPi64, MP-MFLOPSPi64DP, MP-NeonMFLOPS, MP-NeonMFLOPS64

This uses multiply and add calculations with 2 and 32 operations per input data word, using 1, 2, 4 and 8 threads. Data sizes are limited to three to use L1 cache, L2 cache and RAM at 12.8, 128 and 12800 KB (3200, 32000 and 3200000 single precision floating point words or half for double precision). Each thread uses the same calculations but accessing different segments of the data. The program checks for consistent numeric results, primarily to show that all calculations are carried out and can be run.

This benchmark was intended to demonstrate near maximum throughput using single precision floating point calculations. It nearly did on an Intel Core i7 CPU, compiled with gcc under Linux, obtaining 23 out of 32 MFLOPS/MHz with SSE instructions (4 cores, quad word registers, linked multiply and add). The latter arrangement (I believe) also applies to the ARM Cortex-A53 where, with the same efficiency, a Raspberry Pi 3B, at 1200 MHz, would be expected to achieve 27600 MFLOPS and a 3B+ 32200 MFLOPS, at 1400 MHz. For ARM, and probably Intel, as shown below, 20 instructions could be executed at the full speed, with 12 at half speed, nearly corresponding with the 72% (23*100/32) efficiency obtained with Intel.

Single Precision and Double Precision Raspberry Pi 3B+ MFLOPS results are shown below for existing compiled 32 bit and 64 bit benchmarks and one that uses Single Precision NEON Intrinsic functions, then those from a new compilation using gcc 7. None achieve the levels of performance suggested above. Source code and benchmarks for the new MP-MFLOPS, compiled by gcc 7, are in this zip file and this tar.gz file. Speeds of later results from the OpenMP-MFLOPS Benchmark are included in the table.

Performance using one and four threads is shown, along with the gain via the latter. Note that particularly four thread performance can vary significantly, even when using a reliable power supply - See Above.

Raspberry Pi 3B+ MFLOPS at 32 Operations Per Data Word

	32 bit		64 bit		NEON 64 bit	NEON 64 bit gcc7		64 bit OpenMP gcc6	64 bit OpenMP gcc7
	SP	DP	SP	DP	SP	SP	DP	SP	SP
1 Thread	813	798	1793	1405	2999	2800	1403	1692	2781
4 Threads	3189	3109	6981	4398	11563	10608	4492	6469	10007
4T/1T	3.92	3.90	3.89	3.13	3.86	3.79	3.20	3.82	3.60

Source and Assembly Codes for these benchmarks runs are shown below. where explanations of the differences are provided. Next are the detailed results.

3B+ to 3B performance gains are provided following the detailed results. This benchmark tends to be limited by processor speed, producing gains proportional to CPU MHz, but subject to random variations. 64 bit speed gains are also shown (before gcc 7), excluding using RAM, these being greater than 2.1 times single precision and 1.5 times double precision.

```
################## MP-MFLOPS Raspbian RPi 3B 32 Bit #################

   Raspberry Pi 3 CPU 1200 MHz, SDRAM 900 MHz

MP-MFLOPS Linux/ARM V7A v1.0 Sun Jul 15 14:45:33 2018

   FPU Add & Multiply using 1, 2, 4 and 8 Threads

       2 Ops/Word              32 Ops/Word
KB      12.8     128   12800    12.8     128   12800
MFLOPS
1T       184     181     172     697     697     691
2T       367     360     339    1393    1373    1379
4T       642     714     411    2702    2652    2650
8T       597     689     429    2635    2623    2590
Results x 100000
1T     76406   97075   99969   66015   95363   99951

############ RPi 3 V7A2 Double Precision ############

MP-MFLOPS Double Precision v1.0 Sun Jul 15 14:44:57 2018

1T       182     183     160     684     684     670
2T       354     361     216    1365    1342    1320
4T       590     709     215    2695    2695    2544
8T       609     612     219    2576    2662    2529
Results x 100000
1T     76384   97072   99969   66065   95370   99951
```

More Below or Go To Start

################## MP-MFLOPS Raspbian RPi 3B+ 32 Bit ##################

MP-MFLOPS Linux/ARM V7A v1.0 Sun Jul 15 14:04:45 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	214	214	199	813	813	805
2T	420	420	367	1614	1597	1575
4T	739	837	408	3078	3189	3050
8T	741	751	427	3207	3117	3058
Results x 100000						
1T	76406	97075	99969	66015	95363	99951

############ RPi 3B+ Double Precision ##############

MP-MFLOPS Double Precision v1.0 Sun Jul 15 14:03:57 2018

1T	214	213	182	798	783	783
2T	371	424	220	1595	1561	1558
4T	662	808	215	3109	3109	2805
8T	704	813	218	3087	2899	2738
Results x 100000						
1T	76384	97072	99969	66065	95370	99951

################## MP-MFLOPS Gentoo RPi 3B 64 Bit ##################

MP-MFLOPS armv8 64Bit Tue Jul 17 16:01:39 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	711	679	382	1545	1534	1465
2T	1406	1293	384	3088	3054	2838
4T	2468	2554	395	4857	5546	5244
8T	2305	2182	414	5292	5822	5200
Results x 100000						
1T	76406	97075	99969	66015	95363	99951

############## RPi 3 Double Precision ##############

MP-MFLOPS armv8 64Bit Double Precision Tue Jul 17 16:02:25 2018

1T	356	333	177	1211	1188	1075
2T	696	691	208	2412	2413	1926
4T	1251	1272	198	4702	4689	2367
8T	886	1292	196	4514	4267	2715
Results x 100000						
1T	76384	97072	99969	66065	95370	99951

################ MP-MFLOPS Gentoo RPi 3B 64 Bit gcc 7 #################

MP-MFLOPS armv8 64Bit gcc 7 Sun Jun 10 11:45:16 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	709	666	367	2416	2299	2070
2T	1415	1379	357	4721	4763	3670
4T	1663	2362	358	8538	5288	5459
8T	2400	2066	337	9161	7144	5857
Results x 100000						
1T	76406	97075	99969	66015	95363	99951

More Below or Go To Start

############ RPi 3B Double Precision ##############

MP-MFLOPS armv8 64Bit gcc7 DP Sun Jun 10 11:45:45 2018

1T	353	330	150	1206	1188	997
2T	701	687	193	2413	2368	1700
4T	1185	1054	174	4214	3256	2978
8T	1079	1035	173	3240	3541	2993

Results x 100000

1T	76384	97072	99969	66065	95370	99951

################## MP-MFLOPS Gentoo RPi 3B+ 64 Bit ##################

MP-MFLOPS armv8 64Bit Wed Apr 25 10:22:43 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	824	763	380	1793	1795	1704
2T	1620	1577	404	3584	3588	3324
4T	2093	2933	387	6981	6843	4189
8T	2481	2263	409	6851	6391	2944

Results x 100000

1T	76406	97075	99969	66015	95363	99951

############ RPi 3B+ Double Precision ##############

MP-MFLOPS armv8 64Bit Double Precision Wed Apr 25 10:22:59 2018

1T	412	394	185	1405	1380	1275
2T	815	792	200	2769	2801	2402
4T	1468	1244	186	4398	5274	3335
8T	1281	1155	195	4160	5157	3256

Results x 100000

1T	76384	97072	99969	66065	95370	99951

################ MP-MFLOPS Gentoo RPi 3B+ 64 Bit gcc 7 ###############

MP-MFLOPS armv8 64Bit gcc 7 Mon Jun 4 23:58:11 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	814	759	325	2800	2686	2432
2T	1606	1519	288	5566	5570	4011
4T	1814	2822	383	10608	10054	2928
8T	927	874	307	3533	3924	3075

Results x 100000

1T	76406	97075	99969	66015	95363	99951

############ RPi 3B+ Double Precision ##############

MP-MFLOPS armv8 64Bit Double Precision Mon Jun 4 23:54:10 2018

1T	402	383	172	1403	1378	1277
2T	812	795	183	2769	2806	2400
4T	1485	1059	166	4492	5284	3255
8T	1149	1330	171	4420	4961	3221

Results x 100000

1T	76384	97072	99969	66065	95370	99951

################ MP-NeonMFLOPS Raspbian RPi 3B 32 Bit ################

MP-MFLOPS NEON Intrinsics v1.0 Fri Jul 13 18:35:36 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	565	590	424	1760	1723	1679
2T	962	1150	436	3510	3379	3276
4T	1820	1811	435	6869	6625	5989
8T	1645	1990	436	6583	6743	6072

Results x 100000

1T	76406	97075	99969	66014	95363	99951

More Below or Go To Start

################ MP-NeonMFLOPS Raspbian RPi 3B+ 32 Bit ################

MP-MFLOPS NEON Intrinsics v1.0 Sun Jul 15 14:06:03 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	694	686	430	2035	1960	1940
2T	1119	1358	438	4052	3888	3693
4T	1840	2638	438	6911	7827	6384
8T	1881	2540	437	7963	7089	6590
Results x 100000						
1T	76406	97075	99969	66014	95363	99951

################ MP-NeonMFLOPS Gentoo RPi 3B 64 Bit ################

MP-MFLOPS NEON Intrinsics 64 Bit Thu Mar 2 17:03:53 2017

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	12.8	128	12800	12.8	128	12800
MFLOPS						
1T	695	688	386	2595	2577	2423
2T	1373	1242	375	5163	5096	4445
4T	1389	1785	371	10035	10030	6171
8T	2071	2470	339	9410	9481	6209
Results x 100000						
1T	76406	97075	99969	66015	95363	99951

################ MP-NeonMFLOPS Gentoo RPi 3B+ 64 Bit ################

MP-MFLOPS NEON Intrinsics 64 Bit Mon Jun 11 11:59:09 2018

FPU Add & Multiply using 1, 2, 4 and 8 Threads

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
MFLOPS						
1T	788	816	391	2999	2997	2670
2T	1450	1447	414	5986	5900	5057
4T	1593	2126	364	11563	9250	6338
8T	2539	2596	370	9361	11017	5855
Results x 100000						
1T	76406	97075	99969	66015	95363	99951

#################### Compare 3B+ / 3B 1 Thread ####################

	2 Ops/Word			32 Ops/Word		
KB	**12.8**	**128**	**12800**	**12.8**	**128**	**12800**
32 bit SP	1.16	1.18	1.16	1.17	1.17	1.16
32 bit DP	1.18	1.16	1.14	1.17	1.14	1.17
64 bit SP	1.16	1.12	0.99	1.16	1.17	1.16
64 bit DP	1.16	1.16	1.05	1.16	1.16	1.19

############################ 64 bit/32 bit ############################

Pi3B	SP	2.37	3.44	0.73	2.04	2.27	1.79
Pi3B	DP	2.14	1.64	0.67	1.72	1.54	0.95
Pi3B+	SP	2.83	3.50	0.95	2.27	2.15	1.37
Pi3B+	DP	2.22	1.54	0.87	1.41	1.70	1.19

MP-Whetstone Benchmark Next or Go To Start

MP-Whetstone Benchmark - MP-WHETS, MP-WHETSPi64

Multiple threads each run the eight test functions at the same time, but with some dedicated variables. Measured speed is based on the last thread to finish, with Mutex functions, used to avoid the updating conflict by only allowing one thread at a time to access common data. Performance is generally proportional to the number of cores used. There can be some significant differences from the single CPU Whetstone benchmark results on particular tests due to a different compiler being used.

None of the test functions are suitable for SIMD operation, with the simpler instructions being used, possibly leading to some 32 bit tests being faster than those compiled for 64 bits. The Fixed Point MIPS loops are clearly over optimised but, in any case, the time taken has little influence on the overall MWIPS rating.

For both 32 and 64 bit versions, overall single core MWIPS were 17% faster on the 3B+, proportional to CPU MHz ratios. MP speed improvements can be judged by the overall running times shown, which should be similar for 1, 2 and 4 threads and double with 8 threads.

```
################## MP-Whetstone  Raspbian RPi 3B 32 Bit ##################

 MP-Whetstone Benchmark Linux/ARM v1.0 Sun Jun 17 20:55:06 2018

     MWIPS MFLOPS MFLOPS MFLOPS   Cos   Exp    Fixpt      If  Equal
                 1      2      3  MOPS  MOPS    MOPS   MOPS   MOPS

1T   924.2  335.9  276.8  298.5  18.5  10.4  5817.2 1035.3  719.4
2T  1864.8  672.5  664.3  594.1  37.3  20.7 11726.4 2386.9 1438.8
4T  3718.4 1286.3 1303.9 1193.5  74.3  41.5 19961.4 4698.4 2862.7
8T  3908.9 1639.8 1746.6 1274.2  75.9  43.6 29809.6 6321.5 3002.2

  Overall Seconds   5.02 1T,   4.97 2T,   5.07 4T,  10.08 8T

################## MP-Whetstone Raspbian RPi 3B+ 32 Bit ##################

 MP-Whetstone Benchmark Linux/ARM v1.0 Sun Jun 17 22:56:26 2018

     MWIPS MFLOPS MFLOPS MFLOPS   Cos   Exp    Fixpt      If  Equal
                 1      2      3  MOPS  MOPS    MOPS   MOPS   MOPS

1T  1084.2  391.0  384.9  348.6  21.7  12.1  6967.0 1013.1  822.3
2T  2174.4  778.3  775.7  691.9  43.5  24.2 13762.0 2787.4 1675.0
4T  4343.8 1540.9 1558.3 1389.5  86.6  48.4 27529.5 5549.8 3338.5
8T  4548.4 1895.1 1896.0 1504.7  88.0  51.0 39107.6 7287.7 3440.6

  Overall Seconds   5.05 1T,   5.00 2T,   5.06 4T,  10.10 8T

################## MP-Whetstone Gentoo RPi 3B 64 Bit ##################

 MP-Whetstone Benchmark armv8 64 Bit Tue Jun 19 00:00:13 2018

     MWIPS MFLOPS MFLOPS MFLOPS   Cos   Exp     Fixpt      If  Equal
                 1      2      3  MOPS  MOPS     MOPS   MOPS   MOPS

1T   979.8  330.4  322.9  281.5  20.0  10.8 1368033.3  2335.5 1177.1
2T  1986.0  623.9  659.3  564.2  40.0  22.4 2311401.6  4675.1 2355.7
4T  3914.5 1206.2 1295.8 1122.2  78.3  44.3 3007162.3  9230.2 4636.6
8T  4039.7 1498.5 1670.6 1170.2  79.5  45.3 1183764.2 12054.6 5082.7

  Overall Seconds   5.04 1T,   5.01 2T,   5.27 4T,  10.22 8T

################## MP-Whetstone Gentoo RPi 3B+ 64 Bit ##################

 MP-Whetstone Benchmark armv8 64 Bit Tue Jun 26 12:02:45 2018

     MWIPS MFLOPS MFLOPS MFLOPS   Cos   Exp    Fixpt       If  Equal
                 1      2      3  MOPS  MOPS     MOPS    MOPS   MOPS

1T  1151.6  383.0  382.7  327.6  23.2  13.0 1717931.5  2720.5 1364.5
2T  2311.6  766.5  766.8  657.2  46.5  26.0 3478249.3  5460.9 2738.4
4T  4579.6 1505.5 1525.7 1304.4  92.0  51.6 4647842.5 10777.1 5448.5
8T  4788.4 1814.9 1961.4 1381.9  95.0  53.3 5689217.0 13827.3 5810.6

  Overall Seconds   4.96 1T,   4.95 2T,   5.05 4T,  10.07 8T
```

MP-Dhrystone Benchmark Next or Go To Start

MP-Dhrystone Benchmark - MP-DHRYPiA7, MP-DHRYPi64

This runs multiple copies of the whole program at the same time. Dedicated data arrays are used for each thread but there are numerous other variables that are shared. The latter can reduce performance gains via multiple threads and, in certain situations, these can be slower than using a single thread (not in this case).

Average performance gain of the 3B+ over the older 3B were. as usual, the same as the CPU MHz ratio. Single thread performance, at 64 bits, was 55% faster than at 32 bits but, in both cases, reduced to 10% via four threads.

```
################## MP-Dhrystone Raspbian RPi 3B 32 Bit #################

 MP-Dhrystone Benchmark Linux/ARM V7A v1.0 Sun Jun 17 20:36:41 2018

Threads                         1        2        4        8

Seconds                      0.78     0.92     1.27     2.52
Dhrystones per Second     4107750  6949821 10067546 10156278
VAX MIPS rating              2338     3956     5730     5780

        End of test Sun Jun 17 20:36:48 2018


################## MP-Dhrystone Raspbian RPi 3B+ 32 Bit #################

 MP-Dhrystone Benchmark Linux/ARM V7A v1.0 Mon Jun 18 10:05:26 2018

Threads                         1        2        4        8

Seconds                      0.85     0.96     1.36     2.71
Dhrystones per Second     4732954  8293353 11799850 11823294
VAX MIPS rating              2694     4720     6716     6729

        End of test Mon Jun 18 10:05:33 2018


################## MP-Dhrystone Gentoo RPi 3B 64 Bit ##################

 MP-Dhrystone Benchmark armv8 64 Bit Tue Jun 19 00:02:49 2018

Threads                         1        2        4        8

Seconds                      0.63     0.79     1.45     2.86
Dhrystones per Second     6364104 10106501 11050923 11173626
VAX MIPS rating              3622     5752     6290     6359

        End of test Tue Jun 19 00:02:55 2018


################## MP-Dhrystone Gentoo RPi 3B+ 64 Bit ##################

 MP-Dhrystone Benchmark armv8 64 Bit Mon Jun 18 23:11:33 2018

Threads                         1        2        4        8

Seconds                      0.54     0.74     1.24     2.46
Dhrystones per Second     7376153 10819564 12921258 13021546
VAX MIPS rating              4198     6158     7354     7411

        End of test Mon Jun 18 23:11:39 2018
```

MP-Linpack Benchmark Next or Go To Start

MP-Linpack Benchmark - linpackNeonMP, linpackNeonMP64

The original Linpack benchmark for Raspberry Pi, operates on double precision floating point 100x100 matrices (N = 100). This version uses mainly the same C programming code as the single precision floating point NEON compilation. It is run run on 100x100, 500x500 and 1000x1000 matrices using 0, 1, 2 and 4 separate threads. The 0 thread procedures are identical to those in the single core 100 x 100 NEON compilation, using NEON intrinsic functions. The benchmark was produced to demonstrate that the original Linpack 100x100 code could not be converted (by me) to show increased performance using multiple threads. The official line is that users are allowed to use their own linear equation solver for this purpose. These Raspbian tests were carried out under the later Stretch release.

Performance can vary somewhat with this benchmark but reflect the usual 3B+ speed gains, at least on averaging all results. On the same basis, average 64 bit speeds are suggested as being the same as those at 32 bits, but some indicate slower performance. Similar performance could be expected as the compiled code is derived from high level NEON SIMD vector functions.

The poor performance, even using a single thread, is due to the frequent starting and stopping of threads to execute the critical calculations. Consistent threaded speed indicates shared data write back to RAM dependency. This probably increases with larger matrices as more calculations are carried out during a threaded function call.

```
   MFLOPS 0 to 4 Threads, N 100, 500, 1000

################## MP-Linpack  Raspbian RPi 3B 32 Bit ##################

Using NEON Intrinsics, Sun Jun 17 20:32:04 2018

Threads      None        1        2        4

N  100     542.22    61.00    60.67    60.74
N  500     480.55   311.06   316.00   303.48
N 1000     364.07   272.49   231.10   232.07

NR=norm resid RE=resid MA=machep X0=x[0]-1 XN=x[n-1]-1

N              100             500            1000
NR            2.17            5.42            9.50

RE  5.16722466e-05  6.46698638e-04  2.26586126e-03
MA  1.19209290e-07  1.19209290e-07  1.19209290e-07
X0 -2.38418579e-07 -5.54323196e-05 -1.26898289e-04
XN -5.06639481e-06 -4.70876694e-06  1.41978264e-04

################## MP-Linpack Raspbian RPi 3B+ 32 Bit ##################

Using NEON Intrinsics, Mon Jun 18 10:00:08 2018

Threads      None        1        2        4

N  100     633.11    70.82    70.13    70.20
N  500     505.37   323.24   326.81   327.73
N 1000     378.29   337.34   337.01   337.80

SumChecks as above but note 64 bit differences - rounding effects?

################## MP-Linpack Gentoo RPi 3B 64 Bit ##################

 64 Bit NEON Intrinsics, Tue Jun 19 00:04:01 2018

  MFLOPS 0 to 4 Threads, N 100, 500, 1000

Threads      None        1        2        4

N  100     551.48    87.43    81.66    82.68
N  500     359.51   258.43   242.92   255.61
N 1000     296.11   281.75   279.20   282.71

NR=norm resid RE=resid MA=machep X0=x[0]-1 XN=x[n-1]-1

N              100             500            1000
NR            1.97            5.40           13.51

RE  4.69621336e-05  6.44138840e-04  3.22485110e-03
MA  1.19209290e-07  1.19209290e-07  1.19209290e-07
X0 -1.31130219e-05  5.79357147e-05 -3.08930874e-04
XN -1.30534172e-05  3.51667404e-05  1.90019608e-04

################## MP-Linpack Gentoo RPi 3B+ 64 Bit ##################

 64 Bit NEON Intrinsics, Mon Jun 18 23:13:26 2018

Threads      None        1        2        4

N  100     639.82   100.30    95.24    95.25
N  500     430.41   292.80   291.12   290.04
N 1000     349.47   313.59   312.38   313.40

SumChecks as above but note 32 bit differences - rounding effects?
```

MP-BusSpd Benchmark - MP-BusSpeed2PiA7, MP-BusSpeedPi64

This runs integer read only tests using caches and RAM, each thread accessing the same data sequentially. To start with, data is read with large address increments to demonstrate burst data transfers. Performance gains, using L1 cache, can be proportional to the number of cores, but not quite so using L2. The program is designed to produce maximum throughput over buses and demonstrates the fastest RAM speeds using multiple cores.

In the original version, each thread started reading data from the same starting point. This produced acceptable results until shared L2 caches appeared. Then it produced excessive RAM speeds, using more than one thread. With version 2, as used for the following, each thread starts reading from different addresses, providing more realistic results.

Considering just the ReadAll speeds, and MP performance variability, the usual 3B+/3B gains applied. Compared with the BusSpeed benchmark results, the 64 bit one thread performance was much slower and many old 3B cache based speeds significantly faster. In this case, disassembly code was examined to identify why. The ReadAll C code comprises a loop with 64 read statements, using AND. The 64 bit compiler produced code with 64 scalar instructions (e.g. and w3, w3, w0) and 64 loads, compared with 32 bits, with 16 four way SIMD instructions (e.g. vand q15, q15, q6), 16 vector loads, but lots of other adds (for indexing?).

At least, performance on reading data from RAM could be nearly doubled using multithreading.

################## MP-BusSpd Raspbian RPi 3B 32 Bit ##################

MP-BusSpd ARM V7A v2 Fri Jul 13 18:29:45 2018

MB/Second Reading Data, 1, 2, 4 and 8 Threads

KB		Inc32	Inc16	Inc8	Inc4	Inc2	RdAll
12.3	1T	2690	3768	3793	4081	4387	4223
	2T	5086	6856	7148	7710	8571	8159
	4T	8285	11814	13335	15091	16656	15720
	8T	6381	8690	10777	11997	14310	13789
122.9	1T	567	557	1059	1802	2804	3934
	2T	888	903	1746	3287	5379	7686
	4T	895	928	1810	3671	7205	13860
	8T	909	927	1837	3691	7049	13125
12288	1T	120	124	240	475	963	1906
	2T	135	123	245	505	1010	1978
	4T	135	132	259	467	1080	2135
	8T	126	124	255	500	973	2158

End of test Fri Jul 13 18:29:57 2018

################## MP-BusSpd Raspbian RPi 3B+ 32 Bit ##################

MP-BusSpd ARM V7A v2 Fri Jul 13 20:18:36 2018

KB		Inc32	Inc16	Inc8	Inc4	Inc2	RdAll
12.3	1T	3510	4345	4419	4731	5031	4928
	2T	6010	7992	8384	9018	10024	9648
	4T	10127	13748	15247	17581	19516	18252
	8T	7165	10780	13100	14043	16201	16504
122.9	1T	662	648	1247	2090	3246	4565
	2T	1030	1024	2047	3829	6317	8962
	4T	1040	1078	2167	4340	8380	15935
	8T	1052	1077	2122	4263	8362	15826
12288	1T	129	133	267	516	1044	2085
	2T	141	139	280	544	1115	2126
	4T	141	159	301	530	1075	2338
	8T	153	140	273	618	1190	2488

End of test Fri Jul 13 20:18:48 2018

################## MP-BusSpd Gentoo RPi 3B 64 Bit ##################

MP-BusSpd armv8 64 Bit Tue Jun 19 00:06:12 2018

KB		Inc32	Inc16	Inc8	Inc4	Inc2	RdAll
12.3	1T	1462	2407	2584	2038	1461	1492
	2T	4412	4081	4820	3867	2822	2928
	4T	6446	6019	8348	6814	5330	5346
	8T	2598	3924	6114	5788	3827	5016
122.9	1T	535	569	1016	1578	1425	1470
	2T	687	859	1708	3013	2829	2932
	4T	721	878	1829	3573	4369	5261
	8T	780	897	1827	3588	4949	5271
12288	1T	30	111	213	365	835	1024
	2T	45	65	143	337	798	1590
	4T	58	71	253	341	663	1546
	8T	47	97	147	443	904	1821

End of test Tue Jun 19 00:06:25 2018

More Below or Go To Start

################## MP-BusSpd Gentoo RPi 3B+ 64 Bit ##################

MP-BusSpd armv8 64 Bit Tue Jun 26 12:07:50 2018

KB		Inc32	Inc16	Inc8	Inc4	Inc2	RdAll
12.3	1T	3258	2799	3065	2378	1695	1731
	2T	5422	4839	5632	4506	3254	3404
	4T	7974	6313	9724	8025	6234	5524
	8T	5033	4680	6906	6331	5271	5780
122.9	1T	623	666	1188	1871	1657	1694
	2T	1010	1023	2010	3474	3270	3408
	4T	914	1044	2033	4057	6376	6781
	8T	957	1053	2166	4226	5865	6250
12288	1T	115	109	241	484	921	1048
	2T	54	100	227	457	780	2025
	4T	59	99	186	383	868	1623
	8T	67	92	230	481	736	2018

End of test Tue Jun 26 12:08:04 2018

MP-RandMem Benchmark Next or Go To Start

MP-RandMem Benchmark - MP-RandMemPiA7, MP-RandMemPi64

The benchmark has cache and RAM read only and read/write tests using sequential and random access, each thread accessing the same data but starting at different points. It uses the Mutex functions as in Whetstone above, sometimes leading to no performance gains using multiple threads. Although performance via the L1 cache, L2 cache and RAM can be different, it is normally consistent, in each of these areas, during read/write tests.

There can be a lot of variability on 4 thread/1 thread performance gains and many runs might be requires to provide accurate comparisons. On all tests, 3B+/3B performance gains were as expected for cache based results, with averages between 1.6 and 1.7, with RAM performance being similar. Read only MP gains were mainly greater than 3.5 times for cache tests, except on random access to L2, at around 2.4 times, understandably lower using a shared cache. There were also some MP increased throughput using RAM. Raspbian based results indicate slightly improved performance over those using 64 bit Gentoo.

################## MP-RandMem Raspbian RPi 3B 32 Bit ##################

MP-RandMem Linux/ARM v1.0 Sun Jul 15 10:54:39 2018

MB/Second Using 1, 2, 4 and 8 Threads

KB		SerRD	SerRDWR	RndRD	RndRDWR
12.3	1T	4078	3814	4018	3798
	2T	8045	3768	8043	3777
	4T	15622	3724	15625	3730
	8T	15208	3723	15020	3724
122.9	1T	3289	3393	827	891
	2T	6556	3379	1512	880
	4T	12125	3364	2078	886
	8T	12309	3364	2042	886
12288	1T	1669	878	65	64
	2T	3485	872	121	65
	4T	4296	876	146	65
	8T	2435	878	147	65

End of test Sun Jul 15 10:55:24 2018

################## MP-RandMem Raspbian RPi 3B+ 32 Bit ##################

MP-RandMem Linux/ARM v1.0 Sun Jul 15 11:03:26 2018

KB		SerRD	SerRDWR	RndRD	RndRDWR
12.3	1T	4747	4447	4776	4435
	2T	9253	4362	9378	4362
	4T	18114	4343	18080	4322
	8T	17813	4345	17788	4321
122.9	1T	3871	3893	948	1016
	2T	7612	3954	1742	1021
	4T	14399	3929	2383	1025
	8T	14089	3935	2367	1023
12288	1T	1850	860	67	68
	2T	3670	867	126	67
	4T	4097	874	146	68
	8T	2919	873	148	68

End of test Sun Jul 15 11:04:10 2018

################## MP-RandMem Gentoo RPi 3B 64 Bit ##################

MP-RandMem armv8 64 Bit Tue Jun 19 00:08:43 2018

KB		SerRD	SerRDWR	RndRD	RndRDWR
12.3	1T	4260	3071	4261	3081
	2T	7500	3054	7496	3059
	4T	15092	3018	14794	3019
	8T	14315	2977	14544	2989
122.9	1T	3385	2861	867	837
	2T	6323	2653	1543	838
	4T	10638	2873	2009	835
	8T	10810	2841	1947	834
12288	1T	1607	746	71	60
	2T	1605	696	123	59
	4T	1939	766	129	58
	8T	1682	681	141	58

End of test Tue Jun 19 00:09:34 2018

More Below or Go To Start

################## MP-RandMem Gentoo RPi 3B+ 64 Bit ##################

MP-RandMem armv8 64 Bit Tue Jun 26 12:09:25 2018

KB		SerRD	SerRDWR	RndRD	RndRDWR
12.3	1T	4939	3573	4941	3574
	2T	8730	3553	8704	3545
	4T	17121	3499	17197	3498
	8T	16685	3454	17097	3471
122.9	1T	3936	3347	1014	975
	2T	7334	3344	1794	976
	4T	12475	3333	2281	973
	8T	12261	3314	2323	974
12288	1T	1921	793	77	63
	2T	1896	784	113	58
	4T	1621	707	126	58
	8T	1302	764	106	57

End of test Tue Jun 26 12:10:13 2018

OpenMP-MemSpeed Benchmark Next or Go To Start

OpenMP-MemSpeed Benchmark - OpenMP-MemSpeed2, NotOpenMP-MemSpeed2, OpenMP-MemSpeed264, NotOpenMP-MemSpeed264

This is the same as Memory Speed Benchmark, with similar results to NotOpenMP varieties, but with measurements extending to test more memory, also using the OpenMP directive and compile parameter. The NotOpenMP tests use the same code without specifying a compilation using OpenMP. These allow comparisons of MP performance gains over the full range of memory use. There were extremely wide variations in MP performance gains, generally with improvements using data from RAM, but from caches, the best using double precision floating point via Raspbian, and worst all integer tests with a greater than 50% loss.

The usual proportional to MHz 3B+ versus 3B gains were provided, with data from caches, and RAM throughput slightly faster. 64 bit/32 bit NotOpenMP performance ratios were similar to Memory Speed Benchmark, but some were worse using OpenMP. As with some other benchmarks, a new gcc 7 compilation might provide improvement improvement, by including more efficient 64 bit instructions.

############## OpenMP-MemSpeed Raspbian RPi 3B 32 Bit ##############

Memory Reading Speed Test Not OpenMP Version 2 by Roy Longbottom

Start of test Sun Jun 17 20:56:39 2018

Memory	x[m]=x[m]+s*y[m] Int+			x[m]=x[m]+y[m]			x[m]=y[m]		
KBytes	Dble	Sngl	Int32	Dble	Sngl	Int32	Dble	Sngl	Int32
Used	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S
4	1577	2537	3790	2360	3449	3789	2673	2694	2692
8	1594	2547	3811	2388	3469	3812	2717	2716	2716
16	1595	2553	3825	2393	3478	3825	2728	2728	2728
32	1556	2435	3566	2312	3272	3566	2730	2712	2715
64	1508	2300	3304	2177	3065	3303	2542	2485	2485
128	1515	2305	3353	2183	3108	3356	2644	2573	2574
256	1527	2341	3431	2226	3183	3432	2673	2615	2616
512	1406	2083	2869	1983	2702	2873	2558	2495	2404
1024	935	1228	1295	1194	1300	1315	1561	1360	1349
2048	889	1091	1170	1083	1162	1167	1211	1096	1099
4096	890	1109	1169	1089	1167	1168	911	895	903
8192	906	1141	1188	1116	1194	1168	811	804	802
16384	916	1159	1202	1132	1209	1206	766	761	761
32768	928	1166	1206	1119	1224	1206	760	746	746
65536	970	1171	1210	1140	1225	1212	811	810	808
131072	966	1172	1207	1141	1230	1146	953	908	883

End of test Sun Jun 17 20:57:07 2018

Memory Reading Speed Test OpenMP Version 2 by Roy Longbottom

Start of test Mon Jul 9 10:33:36 2018

4	5535	2990	1372	8773	4728	1478	15869	7828	1261
8	6068	3107	1382	10109	5056	1486	16438	8104	1258
16	5739	3119	1317	10193	5114	1393	16624	7862	1220
32	5689	3121	1405	10216	5150	1473	16737	8624	1302
64	5416	3055	1303	8618	4928	1403	12254	8045	1218
128	5396	3050	1359	9101	4932	1379	9496	8089	1249
256	5399	3049	1361	8980	4921	1488	8361	7625	1294
512	4418	2770	1458	6865	4226	1421	5432	5042	1130
1024	3785	2477	1110	4361	3461	1202	1533	1573	1158
2048	3729	2466	975	4268	3439	1200	1017	1017	1150
4096	3714	2477	1144	4228	3370	1431	986	979	1041
8192	3799	2368	1157	3968	3366	1484	961	950	1142
16384	1477	2341	1079	4107	3047	1547	982	985	1037
32768	3351	2499	1080	2089	3216	1437	1005	1001	794
65536	3820	614	1026	3901	3078	1209	1006	1008	954
131072	944	614	746	1160	858	765	1074	1034	566

End of test Mon Jul 9 10:34:05 2018

More Below or Go To Start

############## OpenMP-MemSpeed Raspbian RPi 3B+ 32 Bit ##############

Memory Reading Speed Test Not OpenMP Version 2 by Roy Longbottom

Start of test Mon Jun 18 10:18:27 2018

Memory	x[m]=x[m]+s*y[m] Int+			x[m]=x[m]+y[m]			x[m]=y[m]		
KBytes	Dble	Sngl	Int32	Dble	Sngl	Int32	Dble	Sngl	Int32
Used	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S
4	1839	2961	4423	2755	4024	4423	3115	3143	3140
8	1860	2973	4449	2787	4047	4447	3169	3169	3170
16	1862	2978	4463	2791	4058	4462	3137	3183	3182
32	1789	2759	3988	2612	3684	3979	3188	3180	3179
64	1753	2664	3806	2526	3541	3805	3016	3028	3046
128	1776	2721	3968	2574	3683	3965	3073	2989	2989
256	1797	2768	4066	2621	3779	4068	3108	3045	3045
512	1623	2217	3289	2282	3109	3294	2987	2927	2952
1024	997	1327	1373	1299	1387	1394	1665	1349	1351
2048	942	1142	1175	1133	1171	1173	1204	1054	1050
4096	955	1172	1194	1154	1192	1190	890	886	883
8192	962	1199	1209	1179	1214	1210	802	799	799
16384	976	1216	1217	1194	1214	1216	761	760	759
32768	939	1218	1205	1192	1219	1213	909	864	864
65536	1052	1221	1216	1197	1214	1213	905	870	869
131072	1053	1227	1221	1201	1223	1220	821	800	801

End of test Mon Jun 18 10:18:54 2018

Memory Reading Speed Test OpenMP Version 2 by Roy Longbottom

Start of test Mon Jul 9 10:23:59 2018

4	5851	3452	1602	10192	5484	1719	18419	9062	1469
8	6142	3447	1613	11721	5857	1721	19080	9581	1465
16	6315	3608	1568	11687	5911	1699	19295	9528	1468
32	6292	3606	1451	11704	5926	1617	19413	9486	1396
64	5795	3522	1456	10640	5718	1627	14169	9392	1397
128	5657	3542	1485	10485	5713	1588	11330	9181	1398
256	5769	3501	1395	10164	5624	1597	9837	8853	1387
512	3531	2773	1297	4798	3425	1358	2175	2131	1191
1024	3496	2742	1313	4581	3804	1337	1653	1647	1321
2048	2820	2658	1334	4437	3721	1382	1060	1056	1264
4096	3986	2755	1344	4475	3747	1535	952	1040	1258
8192	4080	2757	1023	4376	3672	1576	1008	992	1147
16384	1434	2385	1212	4418	3500	1481	965	985	1027
32768	1388	2107	1090	4044	3099	1355	941	933	1358
65536	2576	1750	1159	2296	3825	1661	943	971	1509
131072	1035	693	857	1210	918	834	983	967	881

End of test Mon Jul 9 10:24:28 2018

############### OpenMP-MemSpeed Gentoo RPi 3B 64 Bit ###############

Memory Reading Speed Test notOpenMP 64 Bit by Roy Longbottom

Start of test Tue Jun 19 00:12:21 2018

Memory	x[m]=x[m]+s*y[m] Int+			x[m]=x[m]+y[m]			x[m]=y[m]		
KBytes	Dble	Sngl	Int32	Dble	Sngl	Int32	Dble	Sngl	Int32
Used	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S
4	4099	2497	4141	5239	3385	4138	4545	3689	3668
8	3935	2507	4155	5339	3399	4168	4646	3730	3732
16	3857	2506	4171	5358	3417	4155	4680	3744	3751
32	3628	2383	3847	4751	3187	3847	4580	3717	3729
64	3074	2242	3416	3956	2940	3438	4432	3635	3618
128	3042	2274	3495	3933	3008	3492	4209	3488	3498
256	1688	2091	3149	3141	2703	3146	4078	3361	3294
512	550	771	2042	2164	1454	2044	2089	2399	2192
1024	419	984	1199	979	881	1197	1106	1391	1111
2048	448	782	1035	1012	860	1034	764	1061	961
4096	453	941	905	855	1035	1041	833	840	1031
8192	490	882	1082	886	1063	876	772	807	792
16384	478	893	1109	1043	958	966	720	733	734
32768	529	1059	939	1106	975	898	695	701	495
65536	1095	986	1017	990	1104	694	682	686	625
131072	1094	1059	1101	1116	933	1133	608	680	628

End of test Tue Jun 19 00:12:49 2018

More Below or Go To Start

Memory Reading Speed Test OpenMP 64 Bit by Roy Longbottom

Start of test Tue Jun 19 00:14:09 2018

4	5824	3109	1688	8228	4588	1815	11107	5046	2049
8	6064	3205	1690	9310	4858	1824	10003	5704	2052
16	3584	3228	1634	8855	4902	1742	10164	5598	1856
32	2996	3110	1601	9238	4895	1672	9022	5497	1765
64	3697	2665	1563	8723	4798	1655	10674	5331	1758
128	2947	3206	1672	8900	4841	1618	9060	5456	1768
256	2780	2934	1633	9004	4878	1794	8188	5170	1805
512	1048	2730	1750	3995	3993	1593	5896	2950	1298
1024	1114	1912	1580	3900	2488	1205	1272	1258	1025
2048	906	616	1358	3842	3171	1128	855	935	831
4096	658	904	1226	1245	3142	1380	696	756	570
8192	577	965	874	3917	3096	1405	868	742	813
16384	477	2140	914	2886	3074	1003	742	841	896
32768	605	792	1562	1143	2709	1019	899	861	591
65536	1312	555	1163	3974	803	1181	908	833	780
131072	890	572	708	980	819	707	1132	1192	450

End of test Tue Jun 19 00:14:41 2018

############### OpenMP-MemSpeed Gentoo RPi 3B+ 64 Bit ###############

Memory Reading Speed Test notOpenMP 64 Bit by Roy Longbottom

Start of test Mon Jun 18 23:32:35 2018

Memory	x[m]=x[m]+s*y[m] Int+			x[m]=x[m]+y[m]			x[m]=y[m]		
KBytes	Dble	Sngl	Int32	Dble	Sngl	Int32	Dble	Sngl	Int32
Used	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S	MB/S
4	4749	2896	4773	6096	3935	4800	5266	4270	4261
8	4625	2906	4828	6167	3957	4828	5384	4307	4318
16	4624	2902	4840	6203	3954	4838	5422	4347	4332
32	4263	2770	4480	5530	3706	4474	5385	4221	4228
64	3857	2704	4232	4873	3589	4202	5002	4121	4122
128	3899	2760	4371	4906	3720	4373	5030	4079	4111
256	3445	2738	4357	4747	3700	4342	4970	4082	3973
512	760	2098	3027	2577	2271	3004	3830	3325	2786
1024	604	990	1172	1165	1308	1343	979	869	1217
2048	607	940	1160	975	1089	1177	916	887	971
4096	532	908	1017	1155	1156	924	967	1032	1036
8192	579	812	1203	1180	1031	1200	749	861	864
16384	559	1171	1026	1196	1200	1070	727	804	687
32768	498	1107	1064	1197	1202	1135	799	649	763
65536	1176	1060	1165	1177	1203	1230	588	959	963
131072	984	1177	1232	1101	1207	1235	938	723	1047

End of test Mon Jun 18 23:33:03 2018

Memory Reading Speed Test OpenMP 64 Bit by Roy Longbottom

Start of test Mon Jun 18 23:36:51 2018

4	6782	3616	1934	9575	4858	2114	12917	6574	2216
8	7385	3728	1976	8941	5652	2107	13290	6637	2392
16	7099	3769	1945	11092	5751	2105	11593	2798	993
32	1492	1311	781	4563	2391	836	4156	2736	930
64	1432	1264	801	4343	2345	852	3740	2531	914
128	4476	3711	1812	9677	5654	2070	4091	2665	846
256	1085	1563	833	4399	1497	882	3969	2671	894
512	700	615	603	3265	1367	2042	4460	4861	1210
1024	3947	1881	1701	4342	3626	1630	1235	1174	1049
2048	873	1383	1866	1161	3587	1754	915	788	718
4096	670	810	989	1521	3601	1348	944	927	798
8192	808	879	1280	4222	2073	2031	945	927	968
16384	593	1368	1459	4072	1181	1464	959	718	893
32768	777	1994	1471	1223	1269	1800	958	945	790
65536	1736	1275	1271	1006	1249	1787	869	980	1293
131072	988	696	808	1163	928	754	735	1273	562

End of test Mon Jun 18 23:37:24 2018

OpenMP-MFLOPS Benchmark - OpenMP-MFLOPS, notOpenMP-MFLOPS, OpenMP-MFLOPS64, notOpenMP-MFLOPS64

This benchmark carries out the same calculations as the MP-MFLOPS Benchmarks but, in addition, calculations with eight operations per data word. As with OpenMP-MemSpeed, the single core notOpenMP and full OpenMP versions are compiled from the same code and carry out identical numbers of floating point calculations. For some unknown reason, the 32 bit Raspbian versions produce different numeric results at 32 operations per word (see First Results sumcheck).

There are variabilities in measured speeds, but the usual 3B+/3B performance ratios can be assumed. Multiprocessor performance gains were disappointing with the 32 bit Raspbian version, but up to scratch at 64 bits, via Gentoo. The latter benchmark was recompiled using gcc 7 to produce similar best case performance as MP-MFLOPS - see table above. For these particular benchmarks, the only real 64 bit/32 bit gains are on using 32 operations per word (at between 2.3 to 3.0 times).

The gcc 7 versions, **OpenMP-MFLOPS64G7 and notOpenMP-MFLOPS64G7**, can be downloaded from ResearchGate in ompmflops7.tar.gz file.

############## OpenMP-MFLOPS Raspbian RPi 3B 32 Bit ##############

Not OpenMP MFLOPS Benchmark 1 Sun Jun 17 20:58:49 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.763554	655	0.929538	Yes
Data in & out	1000000	2	250	1.206237	415	0.992550	Yes
Data in & out	10000000	2	25	1.134379	441	0.999250	Yes
Data in & out	100000	8	2500	1.161077	1723	0.957126	Yes
Data in & out	1000000	8	250	1.453741	1376	0.995524	Yes
Data in & out	10000000	8	25	1.435932	1393	0.999550	Yes
Data in & out	100000	32	2500	5.024988	1592	0.890268	Yes
Data in & out	1000000	32	250	5.158612	1551	0.988078	Yes
Data in & out	10000000	32	25	5.275346	1516	0.998806	Yes

End of test Sun Jun 17 20:59:12 2018

OpenMP MFLOPS Benchmark 1 Sun Jun 17 21:02:32 2018

Data in & out	100000	2	2500	0.277303	1803	0.929538	Yes
Data in & out	1000000	2	250	1.183362	423	0.992550	Yes
Data in & out	10000000	2	25	1.138538	439	0.999250	Yes
Data in & out	100000	8	2500	0.445954	4485	0.957126	Yes
Data in & out	1000000	8	250	1.299288	1539	0.995524	Yes
Data in & out	10000000	8	25	1.407459	1421	0.999550	Yes
Data in & out	100000	32	2500	4.305910	1858	0.890232	Yes
Data in & out	1000000	32	250	3.822810	2093	0.988068	Yes
Data in & out	10000000	32	25	3.757323	2129	0.998785	Yes

End of test Sun Jun 17 21:02:51 2018

############## OpenMP-MFLOPS Raspbian RPi 3B+ 32 Bit ##############

Not OpenMP MFLOPS Benchmark 1 Mon Jun 18 10:20:24 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.682055	733	0.929538	Yes
Data in & out	1000000	2	250	1.200001	417	0.992550	Yes
Data in & out	10000000	2	25	1.120259	446	0.999250	Yes
Data in & out	100000	8	2500	0.997494	2005	0.957126	Yes
Data in & out	1000000	8	250	1.314719	1521	0.995524	Yes
Data in & out	10000000	8	25	1.262752	1584	0.999550	Yes
Data in & out	100000	32	2500	4.307349	1857	0.890268	Yes
Data in & out	1000000	32	250	4.438297	1802	0.988078	Yes
Data in & out	10000000	32	25	4.432952	1805	0.998806	Yes

End of test Mon Jun 18 10:20:44 2018

More Below or Go To Start

OpenMP MFLOPS Benchmark 1 Mon Jun 18 10:30:42 2018

Data in & out	100000	2	2500	0.230493	2169	0.929538	Yes
Data in & out	1000000	2	250	1.210853	413	0.992550	Yes
Data in & out	10000000	2	25	1.158906	431	0.999250	Yes
Data in & out	100000	8	2500	0.394372	5071	0.957126	Yes
Data in & out	1000000	8	250	1.251015	1599	0.995524	Yes
Data in & out	10000000	8	25	1.199460	1667	0.999550	Yes
Data in & out	100000	32	2500	3.305651	2420	0.890232	Yes
Data in & out	1000000	32	250	3.356769	2383	0.988068	Yes
Data in & out	10000000	32	25	3.325095	2406	0.998785	Yes

End of test Mon Jun 18 10:30:58 2018

################ OpenMP-MFLOPS Gentoo RPi 3B 64 Bit ################

notOpenMP MFLOPS64 Tue Jun 19 00:21:01 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.797313	627	0.929538	Yes
Data in & out	1000000	2	250	1.412364	354	0.992550	Yes
Data in & out	10000000	2	25	1.317844	379	0.999250	Yes
Data in & out	100000	8	2500	1.232307	1623	0.957117	Yes
Data in & out	1000000	8	250	1.658661	1206	0.995518	Yes
Data in & out	10000000	8	25	1.585769	1261	0.999549	Yes
Data in & out	100000	32	2500	5.476343	1461	0.890215	Yes
Data in & out	1000000	32	250	5.663824	1412	0.988088	Yes
Data in & out	10000000	32	25	5.664788	1412	0.998796	Yes

End of test Tue Jun 19 00:21:26 2018

OpenMP MFLOPS64 Tue Jun 19 00:22:01 2018

Data in & out	100000	2	2500	0.256622	1948	0.929538	Yes
Data in & out	1000000	2	250	1.451293	345	0.992550	Yes
Data in & out	10000000	2	25	1.290990	387	0.999250	Yes
Data in & out	100000	8	2500	0.350070	5713	0.957117	Yes
Data in & out	1000000	8	250	1.409139	1419	0.995518	Yes
Data in & out	10000000	8	25	1.279655	1563	0.999549	Yes
Data in & out	100000	32	2500	1.456351	5493	0.890215	Yes
Data in & out	1000000	32	250	1.617333	4946	0.988088	Yes
Data in & out	10000000	32	25	1.596874	5010	0.998796	Yes

End of test Tue Jun 19 00:22:12 2018

################ OpenMP-MFLOPS Gentoo RPi 3B+ 64 Bit ################

notOpenMP MFLOPS64 Mon Jun 18 23:35:18 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.811203	616	0.929538	Yes
Data in & out	1000000	2	250	1.277145	391	0.992550	Yes
Data in & out	10000000	2	25	1.203417	415	0.999250	Yes
Data in & out	100000	8	2500	1.057153	1892	0.957117	Yes
Data in & out	1000000	8	250	1.426572	1402	0.995518	Yes
Data in & out	10000000	8	25	1.384547	1445	0.999549	Yes
Data in & out	100000	32	2500	4.729263	1692	0.890215	Yes
Data in & out	1000000	32	250	4.933636	1622	0.988088	Yes
Data in & out	10000000	32	25	4.928863	1623	0.998796	Yes

End of test Mon Jun 18 23:35:40 2018

More Below or Go To Start

OpenMP MFLOPS64 Mon Jun 18 23:39:10 2018

Data in & out	100000	2	2500	0.242835	2059	0.929538	Yes
Data in & out	1000000	2	250	1.249640	400	0.992550	Yes
Data in & out	10000000	2	25	1.167199	428	0.999250	Yes
Data in & out	100000	8	2500	0.307482	6504	0.957117	Yes
Data in & out	1000000	8	250	1.251838	1598	0.995518	Yes
Data in & out	10000000	8	25	1.157598	1728	0.999549	Yes
Data in & out	100000	32	2500	1.236653	6469	0.890215	Yes
Data in & out	1000000	32	250	1.404484	5696	0.988088	Yes
Data in & out	10000000	32	25	1.357588	5893	0.998796	Yes

End of test Mon Jun 18 23:39:20 2018

############ OpenMP-MFLOPS Gentoo RPi 3B 64 Bit gcc 7 ############

notOpenMP MFLOPS64 GCC7 Mon Jul 9 11:40:27 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.760105	658	0.929538	Yes
Data in & out	1000000	2	250	1.343421	372	0.992550	Yes
Data in & out	10000000	2	25	1.275814	392	0.999250	Yes
Data in & out	100000	8	2500	1.209384	1654	0.957117	Yes
Data in & out	1000000	8	250	1.584924	1262	0.995518	Yes
Data in & out	10000000	8	25	1.570061	1274	0.999549	Yes
Data in & out	100000	32	2500	3.340260	2395	0.890215	Yes
Data in & out	1000000	32	250	3.687632	2169	0.988088	Yes
Data in & out	10000000	32	25	3.684894	2171	0.998796	Yes

End of test Mon Jul 9 11:40:46 2018

OpenMP MFLOPS64 GCC7 Mon Jul 9 11:41:32 2018

Data in & out	100000	2	2500	0.238010	2101	0.929538	Yes
Data in & out	1000000	2	250	1.350017	370	0.992550	Yes
Data in & out	10000000	2	25	1.259776	397	0.999250	Yes
Data in & out	100000	8	2500	0.339368	5893	0.957117	Yes
Data in & out	1000000	8	250	1.353498	1478	0.995518	Yes
Data in & out	10000000	8	25	1.275593	1568	0.999549	Yes
Data in & out	100000	32	2500	0.917181	8722	0.890215	Yes
Data in & out	1000000	32	250	1.406178	5689	0.988088	Yes
Data in & out	10000000	32	25	1.296176	6172	0.998796	Yes

End of test Mon Jul 9 11:41:42 2018

More Below or Go To Start

############ OpenMP-MFLOPS Gentoo RPi 3B+ 64 Bit gcc 7 ############

notOpenMP MFLOPS64 GCC7 Mon Jul 9 11:28:16 2018

Test	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
Data in & out	100000	2	2500	0.646220	774	0.929538	Yes
Data in & out	1000000	2	250	1.199940	417	0.992550	Yes
Data in & out	10000000	2	25	1.158499	432	0.999250	Yes
Data in & out	100000	8	2500	1.049060	1906	0.957117	Yes
Data in & out	1000000	8	250	1.403440	1425	0.995518	Yes
Data in & out	10000000	8	25	1.375663	1454	0.999549	Yes
Data in & out	100000	32	2500	2.876464	2781	0.890215	Yes
Data in & out	1000000	32	250	3.176167	2519	0.988088	Yes
Data in & out	10000000	32	25	3.171387	2523	0.998796	Yes

End of test Mon Jul 9 11:28:33 2018

OpenMP MFLOPS64 GCC7 Mon Jul 9 11:25:46 2018

Data in & out	100000	2	2500	0.250953	1992	0.929538	Yes
Data in & out	1000000	2	250	1.185075	422	0.992550	Yes
Data in & out	10000000	2	25	1.194995	418	0.999250	Yes
Data in & out	100000	8	2500	0.332016	6024	0.957117	Yes
Data in & out	1000000	8	250	1.182464	1691	0.995518	Yes
Data in & out	10000000	8	25	1.157733	1728	0.999549	Yes
Data in & out	100000	32	2500	0.799423	10007	0.890215	Yes
Data in & out	1000000	32	250	1.260969	6344	0.988088	Yes
Data in & out	10000000	32	25	1.196507	6686	0.998796	Yes

End of test Mon Jul 9 11:25:55 2018

Java Benchmark Next or Go To Start

Java Benchmarks

As you probably know, Java programs can run via any Operating System, assuming that a compatible Java RunTime Environment (JRE) is available. The JRE translates a general purpose .class file into hardware dependent computer instructions. The .class files are produced using the javac command from a Java Development Kit (JDK) and these can be run via suitable Operating Systems. In this case, Java Whetstone and JavaDraw benchmarks are considered. Detailed information can be found at ResearchGate in this PDF file.

Java Whetstone Benchmark - whetstc.class

Details of the benchmark are provided above, and include results from the Java version.

JavaDraw Benchmark - JavaDrawPi.class, JavaDrawPC.class

The benchmark uses small to rather excessive simple objects to measure drawing performance in Frames Per Second (FPS). Five tests draw on a background of continuously changing colour shades, each test adding to the load. Further information and download links can be found in the ResearchGate PDF file. Two class files are provided, one produced on a PC, using javac 1.6, and the other on a Raspberry Pi, through javac 1.7. In this case, both have produced similar execution speeds. However, the latter can vary significantly using different Java RTEs, where comparisons can be inappropriate.

As can be seen in the results, the Gentoo 64 bit versions are much slower than those using 32 bit Raspbian, probably a current driver issue. Different drivers and hardware might have also lead to unlike 3B+/3B comparisons, averaging 1.08 times using Raspbian and 1.27 via Gentoo.

```
  Produced by javac 1.7.0_02, run with java 1.8.0_65
  Operating System    Linux, Arch. arm, Version 4.14.34-v7+

################## JavaDraw Raspbian RPi 3B 32 Bit ##################

 Test                               Frames      FPS

 Display PNG Bitmap Twice Pass 1       522    52.19
 Display PNG Bitmap Twice Pass 2       617    61.66
 Plus 2 SweepGradient Circles          627    62.64
 Plus 200 Random Small Circles         603    60.22
 Plus 320 Long Lines                   425    42.44
 Plus 4000 Random Small Circles        306    30.54

        Total Elapsed Time  60.1 seconds

################## JavaDraw Raspbian RPi 3B+ 32 Bit ##################

 Display PNG Bitmap Twice Pass 1       570    56.91
 Display PNG Bitmap Twice Pass 2       663    66.25
 Plus 2 SweepGradient Circles          673    67.29
 Plus 200 Random Small Circles         664    66.38
 Plus 320 Long Lines                   450    44.97
 Plus 4000 Random Small Circles        336    33.51

        Total Elapsed Time  60.1 seconds


Produced by javac 1.7.0_02, run with java 1.8.0_161
Operating System Linux, Arch. aarch64, Version 4.14.44-v8-4fca48b7612d-bis+

################## JavaDraw Gentoo RPi 3B 64 Bit ##################

 Display PNG Bitmap Twice Pass 1       326    32.59
 Display PNG Bitmap Twice Pass 2       529    52.88
 Plus 2 SweepGradient Circles          500    49.97
 Plus 200 Random Small Circles         306    30.55
 Plus 320 Long Lines                    92     9.18
 Plus 4000 Random Small Circles         45     4.46

        Total Elapsed Time  60.2 seconds

################## JavaDraw Gentoo RPi 3B+ 64 Bit ##################

 Display PNG Bitmap Twice Pass 1       391    39.05
 Display PNG Bitmap Twice Pass 2       592    59.18
 Plus 2 SweepGradient Circles          538    53.75
 Plus 200 Random Small Circles         378    37.78
 Plus 320 Long Lines                   167    16.67
 Plus 4000 Random Small Circles         53     5.29

        Total Elapsed Time  60.1 seconds
```

OpenGL GLUT Benchmark - videogl32, videogl64

The first four tests portray moving up and down a tunnel including various independently moving objects, with and without texturing. The last two tests, represent a real application for designing kitchens. The first is in wireframe format, drawn with 23,000 straight lines. The second has colours and textures applied to the surfaces. Note, in 2012, I approved a request from a Quality Engineer at Canonical, to use this OpenGL benchmark in the testing framework of the Unity desktop software, probably as the individual tests can be run for extended periods as Stress Tests, at different screen/window sizes. In this case, the benchmark was run via a script file to show performance using the latter, including a command to turn off VSYNC to allow measured performance to exceed 60 FPS.

The first tests tend to be limited by graphics hardware speed where 3B+/3B comparisons are less than the CPU MHz ratio, with the kitchen tests approaching this 16.7% improvement. Although probably affected be different drivers, 64/32 bit comparisons suggest similar graphics speeds but 64 bit CPU instructions indicated performance gains of more than 30% on the textured kitchen.

As usual, see the original ResearchGate PDF file for more details, links for downloads and other reports.

Example Script File

```
export vblank_mode=0
./videogl32 Width 320, Height 240, NoEnd
./videogl32 Width 640, Height 480, NoHeading, NoEnd
./videogl32 Width 1024, Height 768, NoHeading, NoEnd
./videogl32 NoHeading

NoEnd prevents logging of configuration. Last command uses default resolution.
```

################## OpenGL GLUT Raspbian RPi 3B 32 Bit ##################

GLUT OpenGL Benchmark 32 Bit Version 1, Fri Jul 27 11:56:04 2018

Window Size		Coloured Objects		Textured Objects		WireFrm	Texture
Pixels		Few	All	Few	All	Kitchen	Kitchen
Wide	High	FPS	FPS	FPS	FPS	FPS	FPS
320	240	327.8	191.9	81.6	51.3	21.1	13.4
640	480	245.1	161.1	75.1	48.5	21.0	13.5
1024	768	110.8	102.0	63.8	45.1	21.1	13.4
1920	1080	49.4	47.4	37.0	32.9	20.7	13.2

End at Fri Jul 27 11:58:18 2018

################## OpenGL GLUT Raspbian RPi 3B+ 32 Bit ##################

GLUT OpenGL Benchmark 32 Bit Version 1, Fri Jul 27 11:44:59 2018

320	240	343.2	199.7	88.7	56.6	23.7	15.2
640	480	241.0	168.2	79.9	52.5	23.8	15.1
1024	768	110.5	101.7	63.8	47.1	24.2	15.4
1920	1080	49.7	47.4	36.9	32.8	23.8	15.2

End at Fri Jul 27 11:47:13 2018

################## OpenGL GLUT Gentoo RPi 3B 64 Bit ##################

GLUT OpenGL Benchmark 64 Bit Version 1, Tue Jul 17 19:26:36 2018

160	120	382.3	214.5	118.7	72.3	24.9	18.5
320	240	328.3	199.7	108.9	69.6	24.9	18.4
640	480	220.4	162.2	89.7	62.3	24.9	18.4
1024	768	104.1	96.5	61.1	49.9	24.5	17.9
1920	1080	50.1	47.4	36.6	32.6	23.8	17.7

End at Tue Jul 17 19:29:26 2018

################## OpenGL GLUT Gentoo RPi 3B+ 64 Bit ##################

GLUT OpenGL Benchmark 64 Bit Version 1, Fri Jul 27 11:28:58 2018

160	120	427.2	239.7	132.6	81.3	28.6	21.2
320	240	365.7	224.1	121.5	77.5	28.9	21.3
640	480	247.0	181.6	98.6	68.3	28.5	20.9
1024	768	116.6	107.0	68.5	56.0	28.2	20.7
1920	1080	53.8	51.9	40.3	36.1	27.8	20.5

End at Fri Jul 27 11:31:47 2018

I/O Benchmark Next or Go To Start

I/O Benchmarks

Two varieties of I/O benchmarks are provided, one to measure performance of main and USB drives, and the other for LAN and WiFi network connections. The Raspberry Pi programs write and reads three files at two sizes (defaults 8 and 16 MB), followed by random reading and writing of 1KB blocks out of 4. 8 and 16 MB and finally, writing and reading 200 small files, sized 4, 8 and 16 KB. Run time parameters are provided for the size of large files and file path. The same program code is used for both varieties, the only difference being file opening properties. The drive benchmark includes extra options to use direct I/O, avoiding data caching in main memory, but includes an extra test with caching allowed. For further details and downloads see the usual ResearchGate pdf file.

DriveSpeed Benchmark - DriveSpeed, DriveSpeed64

The first results below access a SanDisk Ultra microSDHC card using Raspbian. With this combination, performance using the Pi 3B+ might be judged as slightly faster (or no different). Next are results from running the Raspbian benchmark on a USB 3 flash drive, faster on large files, but slower on small ones. The df command results, that identify the file path, and run command are also shown.

64 Bit Benchmark - As with earlier attempts to run DriveSpeed64, it failed, providing error reports. This appears to be due to the "do not cache" open file options, as proved by running LanSpeed64 on local drives. Results of the latter are provided below, showing high speed cached data transfers, particularly using default large file sizes. However, large file writing and reading speeds can be measured by specifying much larger files that are too large to be cached in RAM. See the run command and results on another SanDisk Ultra microSDHC card, showing similar data transfer speeds as the other one, but using 0.5 and 1 Mbyte files. Results from random access and small file tests are not influenced by the large file parameter.

```
################## DriveSpeed Raspbian RPi 3B 32 Bit #################

  DriveSpeed RasPi 1.1 Mon Jul 30 14:37:47 2018

Current Directory Path: /home/pi/benchmarks/DriveSpeed
Total MB   14845, Free MB   10483, Used MB    4363

                      MBytes/Second
 MB    Write1    Write2    Write3     Read1     Read2     Read3

  8     18.80     18.81     11.18     23.36     23.45     23.45
 16      8.62     11.26     10.62     23.42     23.49     23.51
Cached
  8   264.48    261.59    272.90    707.81    599.52    753.99

Random          Read                        Write
From MB          4         8        16         4         8        16
msecs       0.323     0.311     0.288      2.56      1.63      1.57

200 Files       Write                       Read                    Delete
File KB          4         8        16         4         8        16     secs
MB/sec        2.36      3.19      2.42      5.96     11.02     12.59
ms/file       1.74      2.57      6.77      0.69      0.74      1.30    0.024

               End of test Mon Jul 30 14:38:18 2018

################## DriveSpeed Raspbian RPi 3B+ 32 Bit #################

  DriveSpeed RasPi 1.1 Mon Jul 30 14:53:47 2018

  8     19.14      6.37     10.66     23.38     23.53     23.61
 16     10.47     10.63     12.90     23.52     23.27     23.60
Cached
  8   226.44    303.78    299.19    547.29    865.25    921.83

Random          Read                        Write
From MB          4         8        16         4         8        16
msecs       0.356     0.401     0.322      1.62      8.13      1.54

200 Files       Write                       Read                    Delete
File KB          4         8        16         4         8        16     secs
MB/sec        2.38      3.01      2.53      8.56      7.86     12.74
ms/file       1.72      2.72      6.48      0.48      1.04      1.29    0.012

               End of test Mon Jul 30 14:54:19 2018
```

More Below or Go To Start

################ USB 3 DriveSpeed Raspbian RPi 3B+ 32 Bit ################

df command

```
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda3       19682912   21792  19661120   1% /media/pi/7CB9-D119
```

Run command

```
./DriveSpeed FilePath /media/pi/7CB9-D119
```

DriveSpeed RasPi 1.1 Mon Jul 30 14:56:56 2018

Selected File Path: /media/pi/7CB9-D119/
Total MB 19222, Free MB 19200, Used MB 21

MBytes/Second

MB	Write1	Write2	Write3	Read1	Read2	Read3
8	17.14	12.09	16.63	37.33	39.62	39.49
16	16.76	12.13	16.34	38.97	39.64	39.63
Cached						
8	55.00	55.95	23.97	883.81	878.40	826.31

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.688	0.685	0.682	2.01	2.05	2.12

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.54	0.88	1.69	5.14	9.67	15.83	
ms/file	7.55	9.34	9.67	0.80	0.85	1.04	0.016

End of test Mon Jul 30 14:57:28 2018

################## DriveSpeed Gentoo RPi 3B+ 64 Bit ##################

LanSpeed RasPi 64 Bit 1.0 Mon Jul 30 16:49:15 2018

Current Directory Path: /home/demouser/benchmarks/driveLANGENTOO
Total MB 28414, Free MB 19958, Used MB 8456

MBytes/Second

MB	Write1	Write2	Write3	Read1	Read2	Read3
8	124.15	162.19	180.79	664.66	683.68	383.51
16	193.74	210.62	184.05	401.49	511.03	456.56

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.003	0.003	0.003	4.03	7.15	3.82

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	41.07	39.81	67.50	255.58	243.20	411.56	
ms/file	0.10	0.21	0.24	0.02	0.03	0.04	0.019

Using Large File Command ./LanSpeed64 MB 512

MBytes/Second

MB	Write1	Write2	Write3	Read1	Read2	Read3
512	16.64	15.55	14.77	19.82	23.54	23.53
1024	17.50	16.24	16.09	20.39	23.54	23.54

LAN Benchmark Next or Go To Start

LAN Benchmark - LanSpeed, LanSpeed64

As indicated above, LanSpeed writes and reads three files at two sizes (defaults 8 and 16 MB), followed by random reading and writing of 1KB blocks out of 4. 8 and 16 MB and finally, writing and reading 200 small files, sized 4, 8 and 16 KB. There are 32 bit and 64 bit versions for the Raspberry Pi, larger varieties for Linux based PCs and a Windows EXE file that can be executed from a remote copy. Details, links to the benchmarks and earlier result can be obtained from ResearchGate in Raspberry Pi 32 Bit and 64 Bit Benchmarks and Stress Tests.pdf.

As shown in the above PDF report, mount statements are required, whereby the benchmarks are run as local programs on the Raspberry Pi and other Linux based systems. Also, Samba was installed to connect the Pi to a Windows Workgroup, in order to run the Intel EXE based benchmark.

The 32 bit benchmarks were run under Raspbian (Stretch) and 64 bit varieties via Gentoo, with a Raspberry Pi 3B+ communicating with a PC running Windows 7 and a dual booted Windows 10/Linux Ubuntu system, also using the older model 3B to Windows 7 to provide comparisons.

Below is a summary of all of the test results that generally provide best case examples. Even so, it is clear that wide variations in performance make it difficult to provide accurate comparisons. Just dealing with the Pi based programs, note the slow random writing speeds to Windows 10. Considering 3B+ to 3B comparisons to Windows 7, later detailed results include some for reading and writing 512 MB files, where the 3B+ is indicated as being 3.3 to 3.4 times faster on reading with 2.2 times improvement on writing. There appears to be some gain on random writing and 200 short file tests, but not much with the smaller data sizes.

16 MB Files MBytes/Second

	Write1	Write2	Write3	Read1	Read2	Read3
Raspbian						
3B >W7	11.42	11.44	11.44	11.67	11.67	11.67
3B+>W7	35.46	36.18	36.22	25.79	25.74	25.59
3B+>W10	35.55	35.95	36.00	26.95	26.95	27.65
3B+>Ubu	34.58	34.46	34.54	27.19	27.29	27.28
W7 >3B+	25.67	25.34	16.71	11.49	8.77	7.09
W10>3B+	25.02	16.97	16.93	11.44	8.66	6.97
Ubu>3B+	27.52	27.49	27.59	38.91	39.01	39.08
Gentoo						
3B >W7	11.22	11.30	11.30	11.63	11.61	11.56
3B+>W7	33.73	35.52	35.19	24.70	22.54	23.39
3B+>W10	33.58	35.30	35.42	13.70	26.50	9.31
3B+>Ubu	33.67	34.73	34.74	20.64	26.90	27.66
W7 >3B+	25.17	23.77	23.82	14.48	10.39	8.05
W10>3B+	17.17	25.31	25.26	14.90	10.56	8.15
Ubu>3B+	21.62	29.01	17.14	39.37	39.62	39.55

Random	**Read milliseconds**			**Write milliseconds**		
From MB	**4**	**8**	**16**	**4**	**8**	**16**
Raspbian						
3B >W7	0.014	0.685	0.829	1.49	1.22	1.35
3B+>W7	0.005	0.659	0.857	0.85	0.91	0.99
3B+>W10	0.005	0.660	1.118	11.79	12.84	14.38
3B+>Ubu	0.005	0.019	0.456	0.49	0.49	0.49
W7 >3B+	0.338	0.335	0.330	0.422	0.422	0.404
W10>3B+	0.471	0.457	0.385	0.474	0.463	0.488
Ubu>3B+	0.49	0.50	0.50			
Gentoo						
3B >W7	0.022	0.746	0.894	1.58	1.43	1.47
3B+>W7	0.024	0.864	0.706	1.09	1.06	1.04
3B+>W10	0.013	0.556	0.775	23.98	15.66	30.23
3B+>Ubu	0.006	0.067	0.552	0.52	0.52	0.52
W7 >3B+	0.617	0.613	0.507	0.694	0.651	0.687
W10>3B+	0.518	0.505	0.499	0.589	0.622	0.609
Ubu>3B+	0.87	0.63	0.61			

200 Files	**Write ms/file**			**Read ms/file**			**Delete**
File KB	**4**	**8**	**16**	**4**	**8**	**16**	**secs**
Raspbian							
3B >W7	4.60	4.42	6.08	2.61	3.22	5.20	0.547
3B+>W7	3.30	3.38	3.68	2.29	2.03	2.71	0.385
3B+>W10	4.49	4.41	4.81	2.15	2.34	2.54	0.274
3B+>Ubu	5.00	5.02	5.33	2.41	2.72	4.19	0.311
W7 >3B+	4.83	4.95	6.01	2.55	2.57	2.95	0.831
W10>3B+	4.74	5.07	5.96	3.20	2.53	3.03	0.841
Ubu>3B+	4.12	5.02	4.96	2.41	2,53	2.83	1.479
Gentoo							
3B >W7	4.78	5.05	6.36	3.07	3.68	6.34	0.833
3B+>W7	3.44	3.88	4.30	2.48	2.16	2.47	0.254
3B+>W10	4.11	4.71	6.81	1.77	2.09	2.84	0.415
3B+>Ubu	5.37	5.56	5.62	3.16	4.32	4.43	0.317
W7 >3B+	4.49	5.37	5.76	2.76	2.73	3.24	0.812
W10>3B+	5.28	5.60	6.21	3.67	3.21	3.52	0.849
Ubu>3B+	3.45	3.69	4.05	2.72	2.72	3.09	1.299

Detailed Results Next or Go To Start

############### Raspbian 32 Bit Detailed Results ###############

Raspberry Pi 3B To Windows 7 PC

LanSpeed RasPi 1.0 Tue May 22 11:43:53 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB   90549, Used MB 175691

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	11.32	11.31	11.38	8.19	11.39	11.61
16	11.42	11.44	11.44	11.67	11.67	11.67

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.014	0.685	0.829	1.49	1.22	1.35

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.89	1.85	2.69	1.57	2.55	3.15	
ms/file	4.60	4.42	6.08	2.61	3.22	5.20	0.547

End of test Tue May 22 11:44:31 2018

================ Raspberry Pi 3B+ To Windows 7 PC ================

LanSpeed RasPi 1.0 Tue May 22 10:36:18 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB   90548, Used MB 175692

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	34.04	8.24	35.16	6.72	26.62	26.68
16	35.46	36.18	36.22	25.79	25.74	25.59

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.005	0.659	0.857	0.85	0.91	0.99

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	1.24	2.42	4.45	1.79	4.04	6.05	
ms/file	3.30	3.38	3.68	2.29	2.03	2.71	0.385

End of test Tue May 22 10:36:46 2018

========================= 512 MB Files Raspbian =========================

	MBytes/Second						Average Gain	
Rpi	Write1	Write2	Write3	Read1	Read2	Read3	Write	Read
3B	11.72	11.72	10.82	11.60	11.62	11.71		
3B+	38.77	38.79	38.67	25.69	25.66	25.55	3.39	2.20

============== Raspberry Pi 3B+ To Windows 10 Core i7 PC ==============

LanSpeed RasPi 1.0 Wed May 23 09:47:34 2018

Selected File Path:
/media/public/
Total MB 346679, Free MB 298782, Used MB   47897

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	33.45	35.57	35.47	3.01	26.49	26.78
16	35.55	35.95	36.00	26.95	26.95	27.65

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.005	0.660	1.118	11.79	12.84	14.38

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.91	1.86	3.40	1.91	3.51	6.44	
ms/file	**4.49**	**4.41**	**4.81**	2.15	2.34	2.54	0.274

End of test Wed May 23 09:48:03 2018

More Below or Go To Start

========== Raspberry Pi 3B+ To Ubuntu Same Core i7 PC ==========

LanSpeed RasPi 1.0 Wed May 23 09:59:57 2018

Selected File Path:
/media/public/
Total MB 446040, Free MB 369385, Used MB   76655

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	34.49	34.64	34.27	27.62	27.61	27.76
16	34.58	34.46	34.54	27.19	27.29	27.28

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.005	0.019	0.456	0.49	0.49	0.49

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.82	1.63	3.07	1.70	3.01	3.91	
ms/file	5.00	5.02	5.33	2.41	2.72	4.19	0.311

End of test Wed May 23 10:00:24 2018

#################### From Remote PC to Raspbian ######################

Windows 7 PC to Raspberry Pi 3B+

CPU AuthenticAMD, Features Code 178BFBFF, Model Code 00100F42
AMD Phenom(tm) II X4 945 Processor Measured 3013 MHz
Windows NT Version 6.1, build 7601, Service Pack 1

Total MB   14845, Free MB   9886, Used MB   4960

LanSpeed Windows 32-Bit Version 1.0, Tue May 22 12:29:16 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	26.20	26.21	25.68	15.02	10.67	8.27
16	25.67	25.34	16.71	11.49	8.77	7.09

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.447	0.435	0.445	0.485	0.488	0.489

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.85	1.65	2.73	1.60	3.19	5.55	
ms/file	4.83	4.95	6.01	2.55	2.57	2.95	0.831

End of test Tue May 22 12:29:49 2018

============== Windows 10 Core i7 PC to Raspberry Pi 3B+ ==============

CPU GenuineIntel, Features Code BFEBFBFF, Model Code 000306E4
Intel(R) Core(TM) i7-4820K CPU @ 3.70GHz Measured 3711 MHz
Windows NT Version 6.2, build 9200,

Total MB   14845, Free MB   9885, Used MB   4960

LanSpeed Windows 32-Bit Version 1.0, Fri May 25 11:30:05 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	5.29	12.15	24.53	14.39	10.25	7.95
16	25.02	16.97	16.93	11.44	8.66	6.97

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.338	0.335	0.330	0.422	0.422	0.404

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.86	1.61	2.75	1.28	3.24	5.41	
ms/file	4.74	5.07	5.96	3.20	2.53	3.03	0.841

End of test Fri May 25 11:30:36 2018

More Below or Go To Start

========== Ubuntu Same Core i7 PC to Raspberry Pi 3B+ ==========

CPU GenuineIntel, Features Code BFEBFBFF, Model Code 000306E4
Intel(R) Core(TM) i7-4820K CPU @ 3.70GHz

Selected File Path:
/media/public/benchmarks/
Total MB 14845, Free MB 10552, Used MB 4293

Linux LAN/WiFi Speed Test 32-Bit Version 1.1, Wed May 23 11:06:39 2018

8 MB File	1	2	3	4	5
Writing MB/sec	27.59	27.20	27.23	27.16	27.22
Reading MB/sec	38.51	38.82	38.95	38.81	38.71

16 MB File	1	2	3	4	5
Writing MB/sec	27.52	27.49	27.59	27.34	27.35
Reading MB/sec	38.91	39.01	39.08	39.18	39.16

32 MB File	1	2	3	4	5
Writing MB/sec	27.73	27.64	27.53	27.81	3.88
Reading MB/sec	39.14	39.20	39.32	39.34	39.33

8 MB Cached File	1	2	3	4	5
Writing MB/sec	27.18	27.19	27.28	27.26	27.27
Reading MB/sec	38.70	38.87	38.85	38.81	38.77

Bus Speed Block KB	64	128	256	512	1024
Reading MB/sec	21315.19	20780.54	17696.24	15950.70	16038.91

1 KB Reads File MB >	2	4	8	16	32	64	128
Random Read msecs	0.50	0.49	0.50	0.50	0.52	0.50	0.50

500 Files	Write		Read		Delete
File KB	MB/sec	ms/File	MB/sec	ms/File	Seconds
2	0.49	4.20	0.87	2.35	1.420
4	0.99	4.12	1.70	2.41	1.381
8	1.63	5.02	3.24	2.53	1.444
16	3.30	4.96	5.79	2.83	1.479
32	5.38	6.09	9.67	3.39	2.806
64	9.50	6.90	14.87	4.41	1.479

End of test Wed May 23 11:08:26 2018

########################## Gentoo 64 bit ############################

Raspberry Pi 3B To Windows 7 PC

LanSpeed RasPi 64 Bit 1.0 Wed May 23 12:45:48 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB 90549, Used MB 175691

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	11.01	11.29	11.23	8.19	11.44	11.48
16	11.22	11.30	11.30	11.63	11.61	11.56

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.022	0.746	0.894	1.58	1.43	1.47

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.86	1.62	2.58	1.34	2.23	2.58	
ms/file	4.78	5.05	6.36	3.07	3.68	6.34	0.833

End of test Wed May 23 12:46:27 2018

More Below or Go To Start

================== Raspberry Pi 3B+ To Windows 7 PC ==================

LanSpeed RasPi 64 Bit 1.0 Wed May 23 17:30:12 2018

Selected File Path:
/media/public/
Total MB 266240, Free MB   90548, Used MB 175692

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	29.76	34.98	35.11	9.04	23.03	26.44
16	33.73	35.52	35.19	24.70	22.54	23.39

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.024	0.864	0.706	1.09	1.06	1.04

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	1.19	2.11	3.81	1.65	3.79	6.63	
ms/file	3.44	3.88	4.30	2.48	2.16	2.47	0.254

End of test Wed May 23 17:30:40 2018

========================= 512 MB Files Gentoo =========================

	MBytes/Second						Average	Gain
Rpi	Write1	Write2	Write3	Read1	Read2	Read3	Write	Read
3B	11.73	11.72	11.72	11.68	11.65	11.70		
3B+	38.84	38.78	38.71	24.98	26.01	26.47	3.31	2.21

============= Raspberry Pi 3B+ To Windows 10 Core i7 PC ============

LanSpeed RasPi 64 Bit 1.0 Wed May 23 16:32:55 2018

Selected File Path:
/media/public/
Total MB 346679, Free MB 298782, Used MB   47897

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	32.10	35.15	34.98	2.49	25.38	7.13
16	33.58	35.30	35.42	13.70	26.50	9.31

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.013	0.556	0.775	23.98	15.66	30.23

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	1.00	1.74	2.40	2.31	3.92	5.78	
ms/file	4.11	4.71	6.81	1.77	2.09	2.84	0.415

End of test Wed May 23 16:33:30 2018

============= Raspberry Pi 3B+ To Ubuntu Same Core i7 PC ============

LanSpeed RasPi 64 Bit 1.0 Wed May 23 17:48:32 2018

Selected File Path:
/media/public/
Total MB 446040, Free MB 369385, Used MB   76655

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	15.34	34.49	34.35	28.15	25.81	27.69
16	33.67	34.73	34.74	20.64	26.90	27.66

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	0.006	0.067	0.552	0.52	0.52	0.52

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.76	1.47	2.92	1.30	1.89	3.70	
ms/file	5.37	5.56	5.62	3.16	4.32	4.43	0.317

End of test Wed May 23 17:49:01 2018

More Below or Go To Start

###################### From Remote PC to Gentoo ######################

Windows 7 PC to Raspberry Pi 3B+

```
 CPU AuthenticAMD, Features Code 178BFBFF, Model Code 00100F42
 AMD Phenom(tm) II X4 945 Processor Measured 3013 MHz
 Windows NT  Version 6.1, build 7601, Service Pack 1

Current Drive Details
Total MB   28414, Free MB   19561, Used MB    8853

LanSpeed Windows 32-Bit Version 1.0, Thu May 24 12:23:40 2018

                      MBytes/Second
 MB   Write1   Write2   Write3    Read1    Read2    Read3

  8    13.30    25.68    25.36    14.91    10.61     8.23
 16    25.17    23.77    23.82    14.48    10.39     8.05

Random         Read                      Write
From MB         4        8       16        4        8       16
msecs       0.617    0.613    0.507    0.694    0.651    0.687

200 Files      Write                     Read                  Delete
File KB         4        8       16        4        8       16    secs
MB/sec       0.91     1.52     2.84     1.48     3.00     5.05
ms/file      4.49     5.37     5.76     2.76     2.73     3.24   0.812

             End of test Thu May 24 12:24:10 2018
```

============== Windows 10 Core i7 PC to Raspberry Pi 3B+ ==============

```
 CPU GenuineIntel, Features Code BFEBFBFF, Model Code 000306E4
 Intel(R) Core(TM) i7-4820K CPU @ 3.70GHz Measured 3711 MHz
 Windows NT  Version 6.2, build 9200,

Current Drive Details
Total MB   28414, Free MB   19593, Used MB    8821

LanSpeed Windows 32-Bit Version 1.0, Thu May 24 11:07:33 2018

                      MBytes/Second
 MB   Write1   Write2   Write3    Read1    Read2    Read3

  8     3.92    24.47    25.34    14.60    10.38     7.81
 16    17.17    25.31    25.26    14.90    10.56     8.15

Random         Read                      Write
From MB         4        8       16        4        8       16
msecs       0.518    0.505    0.499    0.589    0.622    0.609

200 Files      Write                     Read                  Delete
File KB         4        8       16        4        8       16    secs
MB/sec       0.78     1.46     2.64     1.12     2.56     4.66
ms/file      5.28     5.60     6.21     3.67     3.21     3.52   0.849

             End of test Thu May 24 11:08:05 2018
```

More Below or Go To Start

========== Ubuntu Same Core i7 PC to Raspberry Pi 3B+ ==========

Selected File Path:
/media/public/benchmarks/
Total MB 28414, Free MB 20523, Used MB 7892

Linux LAN/WiFi Speed Test 64-Bit Version 1.1, Wed May 23 18:42:55 2018

8 MB File	1	2	3	4	5
Writing MB/sec	29.09	17.24	29.09	16.92	28.89
Reading MB/sec	38.83	38.69	39.35	39.39	39.46

16 MB File	1	2	3	4	5
Writing MB/sec	21.62	29.01	17.14	21.58	7.57
Reading MB/sec	39.37	39.62	39.55	39.61	39.56

32 MB File	1	2	3	4	5
Writing MB/sec	24.96	29.04	24.73	15.63	15.19
Reading MB/sec	39.52	39.73	33.69	39.78	39.81

8 MB Cached File	1	2	3	4	5
Writing MB/sec	17.18	29.09	28.09	17.10	28.53
Reading MB/sec	38.92	39.37	39.34	39.45	39.16

Bus Speed Block KB	64	128	256	512	1024
Reading MB/sec	21712.48	22474.03	18532.30	16844.04	16924.12

1 KB Reads File MB >	2	4	8	16	32	64	128
Random Read msecs	0.78	0.87	0.63	0.61	0.56	0.55	0.55

500 Files	Write		Read		Delete
File KB	MB/sec	ms/File	MB/sec	ms/File	Seconds
2	0.61	3.36	0.76	2.71	1.268
4	1.19	3.45	1.51	2.72	1.282
8	2.22	3.69	3.01	2.72	1.290
16	4.04	4.05	5.30	3.09	1.299
32	3.15	10.40	9.44	3.47	1.296
64	10.81	6.06	14.16	4.63	1.378

End of test Wed May 23 18:44:39 2018

WiFi Next or Go To Start

WiFi Benchmark

These is the same programs as the LAN Benchmarks and run on the same systems, but with the LAN cable disconnected. Running the Pi based benchmark via WiFi, on the Raspberry Pi 3B+, was initially tried under Gentoo, where performance appeared to be more typical of that using a 2.4 GHz hub, or results from the older Pi model 3B. Then I found that my BT Smart Hub had 2.4 and 5 GHz capabilities, with the latter being identified on one of my mobile phones. I moved the Pi closer to the hub, but that made no difference. Then I discovered that the WiFi sometimes needed to be disconnected and reconnected to enable high speed operation. As shown below, there were wide variations is performance, using the Pi and remote systems as host, with some being slower than expectations. At this stage, WiFi no longer worked using my Raspbian Operating System, where results were needed to confirm or contradict the findings.

Numerous suggestions for enabling WiFi were tried, without success. Then, after installation of Raspbian-Stretch, WiFi came alive. As shown in the following results summary, some of the peculiarities also occurred, using Raspbian on the 3B+, including slow reading large files from Windows based programs and slow random writing to Windows 10 but Gentoo failed to achieve 5 GHz type speeds on a number of different tests.

The Pi based benchmark programs were also run on the older model 3B, with relative 3B+ performance gains shown below. Best 3B+/3B performance gains are shown in the detailed results, using larger files and Raspbian, where 3.01 times was shown for writing and 1.60 times fro reading. Performance of random and small file tests was quite similar using Raspbian and Gentoo, but note the highlighted Widows 10 performance in the W10 to 3B+ direction. Then, with the large file tests. the only similarity was 3B+ send (writing to all remote PCs and reading from Ubuntu). In other cases, Gentoo was much slower than the old model 3B on reading. Windows to Raspbian results also suffered from the same problem.

	16 MB Files MBytes/Second						Average Gain	
	Write1	**Write2**	**Write3**	**Read1**	**Read2**	**Read3**	**Write**	**Read**
Raspbian								
3B >W7	4.96	5.07	5.06	5.23	6.76	6.63		
3B+>W7	11.34	13.82	14.14	8.98	9.97	9.77	2.60	1.54
3B+>W10	11.24	13.78	14.19	8.67	10.22	8.57		
3B+>Ubu	11.31	13.30	13.62	10.72	13.25	8.93		
W7 >3B+	9.27	7.34	10.53	5.51	3.92	3.04		
W10>3B+	13.84	13.94	13.77	6.77	4.53	3.41		
Ubu>3B+	12.51	11.27	11.88	14.55	15.53	15.64		
Gentoo								
3B >W7	4.98	5.04	5.11	6.35	6.28	6.11		
3B+>W7	11.54	11.59	11.78	4.27	4.10	4.16	2.31	**0.67**
3B+>W10	10.04	11.00	11.38	3.77	4.19	3.67		
3B+>Ubu	9.94	11.04	11.31	4.21	3.78	4.18		
W7 >3B+	3.91	3.97	3.98	2.75	2.13	1.74		
W10>3B+	4.08	4.06	2.57	2.07	1.76	1.53		
Ubu>3B+	4.19	4.20	4.22	10.24	11.51	11.68		

Random	**Read milliseconds**			**Write milliseconds**			**Average Gain**	
From MB	**4**	**8**	**16**	**4**	**8**	**16**	**Read**	**Write**
Raspbian								
3B >W7	3.360	3.445	3.654	3.64	3.39	3.35		
3B+>W7	2.275	2.755	2.782	2.93	2.68	2.76	1.34	1.24
3B+>W10	10.197	6.838	2.785	**20.95**	**18.85**	**16.46**		
3B+>Ubu	2.429	2.778	2.829	1.39	1.39	1.39		
W7 >3B+	1.375	1.344	1.329	1.570	1.561	1.539		
W10>3B+	1.275	1.262	1.271	1.526	1.495	1.510		
Ubu>3B+	2.11	2.12	2.12					
Gentoo								
3B >W7	3.194	3.472	3.750	3.83	3.59	3.61		
3B+>W7	2.824	2.884	2.964	3.11	2.87	2.90	1.20	1.24
3B+>W10	2.779	2.768	2.740	**20.67**	**20.91**	**20.38**		
3B+>Ubu	2.991	3.160	3.385	1.67	1.61	1.62		
W7 >3B+	1.487	1.421	1.435	1.860	1.779	1.799		
W10>3B+	1.518	1.458	1.400	2.072	2.236	1.980		
Ubu>3B+	2.29	2.29	2.31					

200 Files	**Write ms/file**			**Read ms/file**			**Delete**	**Average Gain**	
File KB	**4**	**8**	**16**	**4**	**8**	**16**	**secs**	**Read**	**Write**
Raspbian									
3B >W7	13.80	15.00	18.23	12.20	16.37	14.66	2.616		
3B+>W7	10.36	11.25	11.22	9.91	10.64	11.11	1.910	1.43	1.37
3B+>W10	11.18	11.91	13.13	10.03	10.52	11.22	1.259		
3B+>Ubu	13.46	13.46	14.05	24.57	11.88	12.19	1.509		
W7 >3B+	12.76	13.15	13.93	4.73	5.83	6.39	1.969		
W10>3B+	**21.54**	**20.53**	**23.41**	6.04	7.48	7.99	2.735		
Ubu>3B+	9.96	10.59	14.38	6.46	7.37	7.68	2.603		
Gentoo									
3B >W7	14.21	17.01	18.07	12.79	15.85	14.83	2.530		
3B+>W7	10.86	11.88	13.69	11.11	12.51	14.31	2.139	1.35	1.15
3B+>W10	11.92	12.94	13.30	10.82	12.00	13.92	2.055		
3B+>Ubu	13.98	14.25	15.05	12.75	13.37	15.74	1.711		
W7 >3B+	14.17	14.91	17.20	5.36	6.35	7.71	2.442		
W10>3B+	**35.64**	**27.95**	**29.61**	7.09	7.78	8.88	3.572		
Ubu>3B+	9.75	11.00	12.77	7.16	7.91	8.87	2.516		

Detailed Results Next or Go To Start

################ Raspbian 32 Bit Detailed Results ################

Raspberry Pi 3B To Windows 7 PC

LanSpeed RasPi 1.0 Mon May 28 10:07:39 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB   90548, Used MB 175692

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	5.08	5.05	5.04	6.13	6.04	6.01
16	4.96	5.07	5.06	5.23	6.76	6.63

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	3.360	3.445	3.654	3.64	3.39	3.35

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.30	0.55	0.90	0.34	0.50	1.12	
ms/file	13.80	15.00	18.23	12.20	16.37	14.66	2.616

End of test Mon May 28 10:08:51 2018

================ Raspberry Pi 3B+ To Windows 7 PC ================

LanSpeed RasPi 1.0 Mon May 28 10:33:49 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB   90547, Used MB 175693

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	11.95	13.23	13.66	9.21	10.59	8.91
16	11.34	13.82	14.14	8.98	9.97	9.77

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	2.275	2.755	2.782	2.93	2.68	2.76

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.40	0.73	1.46	0.41	0.77	1.47	
ms/file	10.36	11.25	11.22	9.91	10.64	11.11	1.910

End of test Mon May 28 10:34:39 2018

================ Raspbian Files 128 MB 3B, 512 MB 3B+ ================

	MBytes/Second						Average Gain	
Rpi	Write1	Write2	Write3	Read1	Read2	Read3	Write	Read
3B	4.62	4.88	4.73	5.71	6.14	5.94		
3B+	14.35	12.93	15.56	9.58	9.42	9.48	3.01	1.60

============== Raspberry Pi 3B+ To Windows 10 Core i7 PC ==============

LanSpeed RasPi 1.0 Tue May 29 10:27:33 2018

Selected File Path:
/media/public/
Total MB 346679, Free MB 298782, Used MB   47897

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	11.84	13.09	13.71	9.57	9.33	11.48
16	11.24	13.78	14.19	8.67	10.22	8.57

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	10.197	6.838	2.785	20.95	18.85	16.46

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.37	0.69	1.25	0.41	0.78	1.46	
ms/file	11.18	11.91	13.13	10.03	10.52	11.22	1.259

End of test Tue May 29 10:28:24 2018

More Below or Go To Start

========== Raspberry Pi 3B+ To Ubuntu Same Core i7 PC ==========

LanSpeed RasPi 1.0 Tue May 29 11:00:24 2018

Selected File Path:
/media/public/
Total MB 446040, Free MB 369312, Used MB   76728

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	12.39	12.81	13.06	11.63	8.20	9.75
16	11.31	13.30	13.62	10.72	13.25	8.93

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	2.429	2.778	2.829	1.39	1.39	1.39

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.30	0.61	1.17	0.17	0.69	1.34	
ms/file	13.46	13.46	14.05	24.57	11.88	12.19	1.509

End of test Tue May 29 11:01:18 2018

#################### From Remote PC to Raspbian ######################

Windows 7 PC to Raspberry Pi 3B+

Current Drive Details
Total MB   14845, Free MB   9892, Used MB   4953

LanSpeed Windows 32-Bit Version 1.0, Tue May 29 11:24:52 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	10.13	10.00	6.91	4.11	3.10	2.51
16	9.27	7.34	10.53	5.51	3.92	3.04

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	1.375	1.344	1.329	1.570	1.561	1.539

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.32	0.62	1.18	0.87	1.40	2.57	
ms/file	12.76	13.15	13.93	4.73	5.83	6.39	1.969

End of test Tue May 29 11:25:41 2018

========== Windows 10 Core i7 PC to Raspberry Pi 3B+ ==========

Current Drive Details
Total MB   14845, Free MB   9892, Used MB   4953

LanSpeed Windows 32-Bit Version 1.0, Tue May 29 11:30:50 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	3.93	13.26	13.58	5.76	3.94	3.04
16	13.84	13.94	13.77	6.77	4.53	3.41

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	1.275	1.262	1.271	1.526	1.495	1.510

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.19	0.40	0.70	0.68	1.10	2.05	
ms/file	21.54	20.53	23.41	6.04	7.48	7.99	2.735

End of test Tue May 29 11:31:45 2018

More Below or Go To Start

============ Ubuntu Same Core i7 PC to Raspberry Pi 3B+ ===========

```
Selected File Path:
/media/public/benchmarks/
Total MB   14845, Free MB   10558, Used MB    4287

Linux LAN/WiFi Speed Test 64-Bit Version 1.1, Tue May 29 11:41:37 2018

               Copyright (C) Roy Longbottom 2011

    8 MB File         1          2          3          4          5
Writing MB/sec      11.58      11.05       9.64       9.03      10.88
Reading MB/sec      10.49      13.18      13.83      14.34      14.87

   16 MB File         1          2          3          4          5
Writing MB/sec      12.53       9.54       8.26       4.38       8.92
Reading MB/sec      13.56      14.95      15.09      15.10      14.99

   32 MB File         1          2          3          4          5
Writing MB/sec      12.51      11.27      11.88      12.18      10.04
Reading MB/sec      14.55      15.53      15.64      15.64      15.69

------------------------------------------------------------------------
8 MB Cached File      1          2          3          4          5
Writing MB/sec      14.98      14.39       8.67      13.08      10.69
Reading MB/sec      10.29      13.50      14.16      14.48      14.73

------------------------------------------------------------------------
Bus Speed Block KB     64        128        256        512       1024
Reading MB/sec   20418.32   20311.32   17283.30   15721.35   15757.56

------------------------------------------------------------------------
1 KB Reads File MB >    2      4      8     16     32     64    128
Random Read msecs    2.09   2.11   2.12   2.12   2.14   2.12   2.09

------------------------------------------------------------------------
500 Files   Write             Read              Delete
File KB     MB/sec  ms/File   MB/sec  ms/File   Seconds
      2       0.21     9.84     0.34     6.08     2.632
      4       0.41     9.96     0.63     6.46     2.671
      8       0.77    10.59     1.11     7.37     2.521
     16       1.14    14.38     2.13     7.68     2.603
     32       2.54    12.89     3.42     9.59     2.717
     64       4.35    15.07     5.41    12.12     2.734

            End of test Tue May 29 11:44:44 2018
```

############################ Gentoo 64 bit ############################

Raspberry Pi 3B To Windows 7 PC

```
  LanSpeed RasPi 64 Bit 1.0 Mon May 28 11:22:59 2018

Selected File Path:
/media/public/ray/
Total MB  266240, Free MB   90548, Used MB  175692

                        MBytes/Second
 MB    Write1    Write2    Write3    Read1     Read2     Read3

  8      4.94      5.06      5.05      6.02      6.31      6.29
 16      4.98      5.04      5.11      6.35      6.28      6.11

Random          Read                        Write
From MB         4         8        16         4         8        16
msecs       3.194     3.472     3.750      3.83      3.59      3.61

200 Files       Write                       Read                    Delete
File KB         4         8        16         4         8        16     secs
MB/sec       0.29      0.48      0.91      0.32      0.52      1.10
ms/file     14.21     17.01     18.07     12.79     15.85     14.83    2.530

               End of test Mon May 28 11:24:12 2018
```

More Below or Go To Start

================= Raspberry Pi 3B+ To Windows 7 PC =================

LanSpeed RasPi 64 Bit 1.0 Mon May 28 11:47:10 2018

Selected File Path:
/media/public/ray/
Total MB 266240, Free MB   90547, Used MB 175693

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	10.42	4.94	11.34	4.27	4.05	4.01
16	11.54	11.59	11.78	4.27	4.10	4.16

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	2.824	2.884	2.964	3.11	2.87	2.90

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.38	0.69	1.20	0.37	0.66	1.14	
ms/file	10.86	11.88	13.69	11.11	12.51	14.31	2.139

End of test Mon May 28 11:48:15 2018

============= Raspberry Pi 3B+ To Windows 10 Core i7 PC ============

LanSpeed RasPi 64 Bit 1.0 Wed May 30 11:50:52 2018

Selected File Path:
/media/public/ray/
Total MB 346679, Free MB 298782, Used MB   47897

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	9.92	11.47	10.15	4.08	3.95	3.98
16	10.04	11.00	11.38	3.77	4.19	3.67

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	2.779	2.768	2.740	20.67	20.91	20.38

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.34	0.63	1.23	0.38	0.68	1.18	
ms/file	11.92	12.94	13.30	10.82	12.00	13.92	2.055

End of test Wed May 30 11:51:58 2018

============= Raspberry Pi 3B+ To Ubuntu Same Core i7 PC ============

LanSpeed RasPi 64 Bit 1.0 Mon May 28 12:21:33 2018

Selected File Path:
/media/public/
Total MB 446040, Free MB 369282, Used MB   76758

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	10.23	10.89	10.51	3.92	4.20	4.16
16	9.94	11.04	11.31	4.21	3.78	4.18

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	2.991	3.160	3.385	1.67	1.61	1.62

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.29	0.57	1.09	0.32	0.61	1.04	
ms/file	13.98	14.25	15.05	12.75	13.37	15.74	1.711

End of test Mon May 28 12:22:39 2018

More Below or Go To Start

###################### From Remote PC to Gentoo ######################

Windows 7 PC to Raspberry Pi 3B+

Current Drive Details Total MB 28414, Free MB 18979, Used MB 9435

LanSpeed Windows 32-Bit Version 1.0, Wed May 30 11:45:22 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	4.09	4.08	4.02	2.72	2.09	1.72
16	3.91	3.97	3.98	2.75	2.13	1.74

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	1.487	1.421	1.435	1.860	1.779	1.799

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.29	0.55	0.95	0.76	1.29	2.13	
ms/file	14.17	14.91	17.20	5.36	6.35	7.71	2.442

End of test Wed May 30 11:46:31 2018

============== Windows 10 Core i7 PC to Raspberry Pi 3B+ =============

Current Drive Details Total MB 28414, Free MB 18979, Used MB 9435

LanSpeed Windows 32-Bit Version 1.0, Wed May 30 11:01:59 2018

	MBytes/Second					
MB	Write1	Write2	Write3	Read1	Read2	Read3
8	4.23	4.20	4.10	2.82	2.23	1.83
16	4.08	4.06	2.57	2.07	1.76	1.53

Random	Read			Write		
From MB	4	8	16	4	8	16
msecs	1.518	1.458	1.400	2.072	2.236	1.980

200 Files	Write			Read			Delete
File KB	4	8	16	4	8	16	secs
MB/sec	0.11	0.29	0.55	0.58	1.05	1.84	
ms/file	35.64	27.95	29.61	7.09	7.78	8.88	3.572

End of test Wed May 30 11:03:20 2018

============== Ubuntu Same Core i7 PC to Raspberry Pi 3B+ ============

Selected Path /media/public/benchmarks/ MB 28414, Free 20522, Used 7892

Linux LAN/WiFi Speed Test 64-Bit Version 1.1, Mon May 28 12:27:59 2018

8 MB File	1	2	3	4	5
Writing MB/sec	3.96	4.08	4.21	3.36	4.13
Reading MB/sec	9.43	11.46	11.51	11.49	11.68

16 MB File	1	2	3	4	5
Writing MB/sec	4.19	4.20	4.22	4.19	4.13
Reading MB/sec	10.24	11.51	11.68	11.92	11.88

32 MB File	1	2	3	4	5
Writing MB/sec	4.19	4.16	4.14	3.46	4.18
Reading MB/sec	10.96	11.59	11.87	12.16	12.64

8 MB Cached File	1	2	3	4	5
Writing MB/sec	4.14	4.11	4.11	4.16	4.11
Reading MB/sec	11.21	11.60	11.81	11.69	11.77

Bus Speed Block KB	64	128	256	512	1024
Reading MB/sec	21401.17	21459.99	18396.83	16737.76	16867.78

1 KB Reads File MB >	2	4	8	16	32	64	128
Random Read msecs	2.27	2.29	2.29	2.31	2.31	2.31	2.31

500 Files	Write		Read		Delete
File KB	MB/sec	ms/File	MB/sec	ms/File	Seconds
2	0.22	9.51	0.30	6.82	2.648
4	0.42	9.75	0.57	7.16	2.606
8	0.74	11.00	1.04	7.91	2.650
16	1.28	12.77	1.85	8.87	2.516
32	2.03	16.17	3.12	10.50	2.535
64	2.71	24.17	4.91	13.35	2.537

Stress Tests Next or Go To Start

Stress Tests

The stress test programs have run time parameters that control duration, often with selection of an available test function and/or data size. Performance results are displayed and logged in text files as the benchmarks are running, and a specific output sampling frequency might be available.

During the tests, another program was available to measure CPU MHz and core temperature, on a sampling basis. In view of voltage related problems identified during MultiThreading Benchmarks, measurement of this has been included in new 32 bit and 64 bit versions. As shown below, the voltage option can be of importance in considering heating effects on performance.

MHz, Temperature and Core Voltage Monitor - RPiHeatMHzVolts, RPiHeatMHzVolts64G

These new programs are available in this ResearchGate tar.gz file. An example of run time parameters are shown below, where seconds are the intervals between sample measurements. Because of the coarse sampling, changes in readings are not necessarily synchronised. As shown in this Raspberry Pi Report, Model 3B+ thermal control has changed from that used with the 3B. Above 70°C, core frequency is reduced from 1.4 GHz to 1,2 GHz, when core voltage is reduced. Then thermal throttling is applied on reaching 80°C. These characteristics are reflected in the following results, measured whilst running four copies of the Integer Stress Test, reported in the next section.

```
Command  ./RPiHeatMHzVolts64G passes 60, seconds 16

Temperature and CPU MHz Measurement  Start at Tue Jul 31 21:14:36 2018

Seconds
   0.0     1400 scaling MHz,   1400 ARM MHz, core volt=1.3438V, temp=58.0°C
  16.0     1400 scaling MHz,   1400 ARM MHz, core volt=1.3500V, temp=65.0°C
  32.5     1400 scaling MHz,   1400 ARM MHz, core volt=1.3563V, temp=69.3°C
  49.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.3563V, temp=70.4°C
  65.6     1400 scaling MHz,   1199 ARM MHz, core volt=1.2375V, temp=70.9°C
  82.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=71.4°C
  98.7     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=72.0°C
 115.2     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=73.1°C
 131.7     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=74.1°C
 148.3     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=74.1°C
 164.9     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=74.7°C
 181.5     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=75.2°C
 197.9     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=75.2°C
 214.4     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=75.2°C
 230.9     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=76.3°C
 247.5     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 264.0     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 280.5     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 297.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 313.6     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 330.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=77.4°C
 346.7     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=78.4°C
 363.2     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=78.4°C
 379.8     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=78.4°C
 396.4     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 413.0     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 429.6     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 446.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 462.6     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 479.2     1400 scaling MHz,   1195 ARM MHz, core volt=1.2375V, temp=80.1°C
 495.9     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 512.3     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 528.8     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 545.3     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 561.8     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=78.4°C
 578.3     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.0°C
 594.9     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=79.5°C
 611.6     1400 scaling MHz,   1195 ARM MHz, core volt=1.2375V, temp=79.5°C
 628.1     1400 scaling MHz,   1195 ARM MHz, core volt=1.2375V, temp=79.5°C
 644.6     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=80.6°C
 661.1     1400 scaling MHz,   1200 ARM MHz, core volt=1.2375V, temp=80.6°C
 677.6     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.1°C
 694.2     1400 scaling MHz,   1194 ARM MHz, core volt=1.2375V, temp=80.6°C
 710.7     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 727.3     1400 scaling MHz,   1195 ARM MHz, core volt=1.2375V, temp=80.6°C
 743.8     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 760.3     1400 scaling MHz,   1087 ARM MHz, core volt=1.2375V, temp=80.1°C
 776.8     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 793.3     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 809.8     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 826.3     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=81.1°C
 842.9     1400 scaling MHz,   1087 ARM MHz, core volt=1.2375V, temp=80.6°C
 859.5     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 876.0     1400 scaling MHz,   1087 ARM MHz, core volt=1.2375V, temp=81.1°C
 892.5     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 909.1     1400 scaling MHz,   1140 ARM MHz, core volt=1.2375V, temp=80.6°C
 925.6     1400 scaling MHz,   1087 ARM MHz, core volt=1.2375V, temp=81.1°C
 942.2     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=80.6°C
 958.7     1400 scaling MHz,   1141 ARM MHz, core volt=1.2375V, temp=81.1°C
```

Integer Stress Tests - stressIntPiA7, stressIntPi64 - Four Copies

These have six write/read and six read only tests, with different variations of data patterns. The read phase comprises an equal number of additions and subtractions, with the data being unchanged afterwards and checked for correctness. Speed is measured in MB/second. Results are displayed at 10 second intervals. Run time parameters are provided for KBytes memory used, seconds for each of the twelve tests and log number for use in multitasking. The 32 bit version can be run from a shell script, as shown below, with lxterminal commands to use different terminal windows. This is not available (or not found for installation), at 64 bits in Gentoo, separate terminals needing to be opened for individual execute commands. The 32 bit benchmark can be obtained from Raspberry_Pi_2_Stress_Tests.zip, and the 64 bit version in Rpi3-64-Bit-Benchmarks.tar.gz.

Next, in the table, are the first to start results from a 64 bit Gentoo session that reported the above MHz, voltage and temperature measurements (started 3 seconds earlier). The test program is intended to run each part for the same number of seconds, leading to lower pass counts as the CPU speed reduces. This is followed by performance obtained on running a single copy of the benchmarks with L1 cache, L2 cache and RAM based data. In this case, the 32 bit versions are shown to be faster than the 64 bit compilations.

Finally are a range of 3B+ stress tests results, where no heat sink was used and it was installed in a plastic case. The cool ones were from first runs when the room temperature was 23 °C, and the hot ones from a second test when the room temperature was 3 to 4 °C higher. The critical 80 °C was breached in all cases, but not until the Read Only section with the cooler tests. Performance degradation is shown to be quite similar at 32 and 64 bits, at least with Read/Write tests, also on comparing MB/second and CPU MHz reductions.

Shell Script

```
lxterminal --geometry=80x15 -e ./RPiHeatMHzVolts passes 60, seconds 16
lxterminal --geometry=80x15 -e ./stressIntPiA7 KB 16 Secs 80 Log 21
lxterminal --geometry=80x15 -e ./stressIntPiA7 KB 16 Secs 80 Log 22
lxterminal --geometry=80x15 -e ./stressIntPiA7 KB 16 Secs 80 Log 23
lxterminal --geometry=80x15 -e ./stressIntPiA7 KB 16 Secs 80 Log 24
```

```
  Gentoo Integer Stress Test RPi 64 Tue Jul 31 21:14:39 2018

 16 KBytes Cache or RAM Space, 80 Seconds Per Test, 12 Tests

Write/Read
 1    2748 MB/sec  Pattern 00000000 Result OK    6708355 passes
 2    2529 MB/sec  Pattern FFFFFFFF Result OK    6173859 passes
 3    2504 MB/sec  Pattern A5A5A5A5 Result OK    6113617 passes
 4    2508 MB/sec  Pattern 55555555 Result OK    6124102 passes
 5    2515 MB/sec  Pattern 33333333 Result OK    6141199 passes
 6    2504 MB/sec  Pattern F0F0F0F0 Result OK    6113284 passes
Read
 1    2806 MB/sec  Pattern 00000000 Result OK   13702300 passes
 2    2826 MB/sec  Pattern FFFFFFFF Result OK   13797100 passes
 3    2740 MB/sec  Pattern A5A5A5A5 Result OK   13378700 passes
 4    2676 MB/sec  Pattern 55555555 Result OK   13068900 passes
 5    2656 MB/sec  Pattern 33333333 Result OK   12967300 passes
 6    2658 MB/sec  Pattern F0F0F0F0 Result OK   12977800 passes
```

Single Core Speeds, 32 Bit Raspbian and 64 Bit Gentoo

	Write/Read MB/scond						Read					
	32 Bit			64 Bit			32 Bit			64 Bit		
KB	16	64	2048	16	64	2048	16	64	2048	16	64	2048
3B+	3883	3786	1681	2991	2910	1480	4246	3625	1907	3344	2985	1800

##

	Raspbian 32 Bit						Gentoo 64 Bit					
	Cool			Hot			Cool			Hot		
	MB/s	MHz	°C	MB/s	MHz	°C	MB/s	MHz	°C	MB/s	MHz	°C
Write/Read		1400	44.0		1400	59.1		1400	47.8		1399	65.5
1	3699	1400	68.2	3240	1200	75.2	2927	1400	69.8	2589	1200	78.4
2	3415	1200	70.9	3297	1200	79.0	2611	1200	70.9	2509	1141	81.1
3	3218	1200	72.9	3247	1195	80.6	2536	1200	72.0	2375	1087	81.7
4	3299	1200	73.1	3045	1141	81.1	2525	1200	75.2	2302	1034	81.7
5	3288	1200	75.2	2929	1087	81.7	2533	1200	75.8	2263	1034	81.7
6	3291	1200	76.3	2882	1033	81.7	2533	1200	77.4	2231	1034	81.7
Read												
1	3620	1200	75.8	3343	1141	81.7	2832	1200	77.4	2646	1141	81.7
2	3602	1200	78.4	3153	1034	81.7	2841	1200	78.4	2539	1034	81.7
3	3592	1195	79.5	3015	980	82.2	2829	1195	80.1	2470	1034	81.7
4	3567	1141	80.5	2938	926	82.2	2790	1141	80.6	2444	1033	82.2
5	3500	1141	80.6	2922	980	82.7	2733	1141	80.6	2414	1034	82.2
6	3432	1087	80.6	2876	980	82.2	2679	1087	80.1	2152	980	82.2
1 Core RW	3883	1400		3883	1400		2991	1400		2991	1400	
1 Core Rd	4246	1400		4246	1400		3344	1400		3344	1400	
MIn RW	3218	1200		2882	1033		2525	1200		2231	1034	
Min Rd	3432	1087		2876	926		2679	1087		2152	980	
%1 Core RW	83	86		74	74		84	86		75	74	
%1 Core Rd	81	78		68	66		80	78		64	70	

Floating Stress Tests Next or Go To Start

Floating Point Stress Tests - burninfpuPi2, burninfpuPi64 - Four Copies

These use the same program test code as MP-MFLOPS Benchmark, but just for a single CPU core, and the loop for repeat passes included in the main functions, to enable faster execution (up to 17.2 GFLOPS with 4 cores, compared with 11.6). The arithmetic operations executed are of the form x[i] = (x[i] + a) * b - (x[i] + c) * d + (x[i] + e) * f with 2, 8 or 32 operations per input data word. The same variables are used for each word and final results are checked for consistency, any errors being reported. The benchmark has input parameters for KWords, Section 1, 2 or 3 (for 2, 8 or 32 operations per word) and log number (0 to 99). The 32 bit benchmark can be obtained from Raspberry_Pi_2_Stress_Tests.zip, and the 64 bit version in Rpi3-64-Bit-Benchmarks.tar.gz.

Below is the start of the program output where, in this case, each pass carries out the same number of calculations, resulting in longer time when the core MHz reduces. This is followed by performance, in MFLOPS, for the three increasing operations per word, using L1 cache, L2 cache and RAM, via both 32 bit and 64 bit compilations. Note that the latter obtains the highest maximum speeds, but is slower using certain test functions.

No Heatsink

The next table provides results of fifteen minute stress tests, on the Raspberry Pi 3B+, in a plastic case and no heatsink, using 32 bit Raspbian and 64 bit Gentoo. In this case, the cool runs followed immediately powering on with room temperatures around 23°C and the hot ones shortly after the others finished. With the different timing procedures, the MFLOPS, MHz and temperature measurements shown are at approximately the same time. MFLOPS is the average speed over 15 to 17 minutes, temperature and MHz instantaneously at around 16 minute intervals, with both varying up and down within a test.

Although the MFLOPS speed is slower, the 32 bit test appears to generate higher temperatures, with earlier degradation to 1200 MHz and further reductions on exceeding 80°C (worst cases 81.7°C, 1033 MHz, 2681 from 3439 MFLOPS). The 32 bit version was compiled to use NEON four way SIMD instructions, with the other employing the more recent 64 bit SIMD vector functions, apparently less demanding power wise.

Part example one of four 64 bit programs

Command ./burninfpuPi64 KWords 4, Section 2, minutes 15, Log 21

Burn-In-FPU RPi 64 Sat Aug 11 13:04:10 2018

Using 16 KBytes, 8 Operations Per Word, For Approximately 15 Minutes

Pass	4 Byte Words	Ops/ Word	Repeat Passes	Seconds	MFLOPS	First Results	All Same
1	4000	8	1996000	15.03	4251	0.539296687	Yes
2	4000	8	1996000	15.56	4104	0.539296687	Yes
3	4000	8	1996000	16.58	3851	0.539296687	Yes
4	4000	8	1996000	17.16	3721	0.539296687	Yes
5	4000	8	1996000	17.40	3671	0.539296687	Yes
6	4000	8	1996000	17.74	3600	0.539296687	Yes
7	4000	8	1996000	17.87	3574	0.539296687	Yes
8	4000	8	1996000	17.72	3605	0.539296687	Yes
9	4000	8	1996000	17.59	3630	0.539296687	Yes
10	4000	8	1996000	17.55	3640	0.539296687	Yes

Single Core MFLOPS, 32 Bit Raspbian and 64 Bit Gentoo

Ops/Word	2	2	2	8	8	8	32	32	32
K Bytes	16	64	2048	16	64	2048	16	64	2048
K Words	4	16	512	4	16	512	4	16	512
32 Bit	1788	1672	413	3439	3365	1636	2011	2000	1846
64 Bit	2070	1924	405	4360	4278	1617	1781	1775	1696

More Below or Go To Start

Pi 3B+ Floating Point Stress Tests, 16 KBytes, 8 Operations Per Word - No Heatsink

	Raspbian 32 Bit						Gentoo 64 Bit					
	Cool			Hot			Cool			Hot		
Secs	MFLOPS	MHz	°C	MFLOPS	MHz	°C	MFLOPS	MHz	°C	MFLOPS	MHz	°C
0		1400	47.8		1400	55.8		1400	45.1		1400	61.2
16	3396	1400	62.8	3257	1200	69.8	4315	1400	54.2	4289	1400	67.7
33	3377	1400	67.1	2960	1199	70.4	4336	1400	59.6	3755	1200	70.9
49	3400	1400	69.8	2844	1200	70.9	4343	1400	62.3	3667	1200	71.4
66	3322	1199	69.8	2839	1200	71.4	4311	1400	64.5	3689	1200	72.5
82	3181	1400	70.4	2838	1200	73.1	4296	1400	65.5	3692	1200	73.1
99	3100	1200	70.4	2845	1200	73.6	4295	1400	68.8	3673	1200	74.1
116	3028	1200	70.9	2865	1200	74.1	4319	1400	69.3	3671	1199	74.1
132	3003	1200	70.9	2926	1200	75.2	4291	1200	69.8	3669	1200	74.1
149	2939	1200	70.9	2934	1200	75.2	4150	1400	69.8	3686	1200	75.2
166	2942	1200	70.9	2927	1200	75.8	4058	1200	70.9	3697	1200	75.8
182	2940	1200	70.9	2926	1200	76.3	4012	1200	69.8	3660	1200	75.8
199	2942	1200	71.4	2902	1200	76.8	3977	1200	70.9	3662	1199	76.3
215	2931	1200	72.0	2914	1200	76.8	3927	1200	70.9	3676	1200	76.3
232	2929	1200	72.5	2888	1200	76.3	3898	1200	70.9	3678	1200	76.8
249	2913	1200	73.1	2890	1200	77.4	3689	1200	70.4	3679	1200	76.3
265	2913	1200	73.1	2917	1200	77.9	3712	1200	70.4	3692	1200	77.4
282	2933	1200	73.1	2899	1200	77.9	3823	1199	70.9	3630	1200	76.8
299	2935	1200	73.6	2892	1200	77.9	3799	1199	70.9	3688	1199	76.8
315	2934	1200	74.1	2911	1200	78.4	3783	1200	70.9	3687	1200	77.4
332	2931	1200	74.1	2912	1200	79.5	3733	1200	70.4	3688	1200	77.4
349	2829	1200	74.1	2898	1200	79.0	3747	1200	70.9	3675	1200	76.8
365	2860	1200	74.7	2907	1200	79.5	3715	1200	70.9	3678	1200	77.9
382	2932	1200	74.7	2912	1200	79.5	3705	1199	70.9	3691	1200	76.8
399	2928	1200	75.2	2893	1200	79.5	3686	1200	71.4	3690	1200	77.9
415	2930	1200	75.2	2906	1200	80.1	3698	1199	72.0	3686	1200	78.4
432	2926	1200	75.2	2893	1195	79.5	3703	1200	71.4	3692	1200	77.9
448	2927	1200	75.8	2881	1200	80.1	3689	1200	71.4	3691	1200	78.4
465	2938	1200	76.3	2863	1141	80.1	3680	1200	72.0	3666	1200	77.9
482	2937	1200	76.3	2875	1195	79.5	3686	1200	72.0	3659	1200	78.4
498	2933	1200	76.8	2857	1200	80.6	3685	1200	72.0	3669	1200	78.4
515	2935	1200	76.3	2834	1141	80.6	3709	1200	73.1	3667	1200	78.4
532	2919	1200	77.4	2840	1141	80.6	3688	1200	72.5	3691	1200	78.4
548	2914	1200	77.4	2802	1195	80.6	3692	1200	73.1	3638	1200	79.5
565	2922	1200	77.4	2815	1195	80.6	3686	1200	72.5	3655	1200	78.4
581	2839	1200	77.4	2815	1141	80.1	3699	1200	73.1	3645	1200	78.4
598	2862	1200	77.4	2810	1141	80.1	3665	1200	73.1	3674	1200	79.0
615	2840	1200	77.9	2805	1141	80.6	3652	1200	73.6	3671	1200	78.4
631	2862	1200	78.4	2809	1141	80.6	3700	1200	73.6	3684	1200	78.4
648	2865	1200	78.4	2791	1195	80.6	3704	1200	73.6	3678	1200	79.0
665	2844	1200	79.0	2758	1141	80.6	3648	1200	74.1	3683	1200	79.0
681	2790	1200	77.9	2764	1141	80.6	3697	1200	74.1	3683	1200	78.4
698	2849	1200	79.5	2765	1141	80.6	3605	1200	74.1	3693	1200	79.0
714	2927	1200	78.4	2768	1141	80.6	3684	1200	73.6	3654	1200	79.5
731	2929	1200	79.5	2758	1141	80.6	3680	1200	74.1	3655	1200	79.5
748	2908	1200	79.5	2777	1141	80.6	3647	1200	74.1	3642	1200	79.0
764	2898	1200	79.5	2775	1141	80.6	3695	1200	74.1	3693	1200	79.0
781	2903	1200	79.5	2746	1141	80.6	3704	1200	74.1	3602	1195	79.5
797	2911	1200	79.0	2767	1141	80.6	3674	1199	75.2	3693	1200	79.5
814	2884	1199	79.0	2758	1141	80.6	3679	1200	75.2	3693	1195	79.5
830	2898	1195	79.5	2741	1141	80.6	3689	1200	75.2	3689	1200	79.0
847	2880	1200	79.5	2744	1140	80.6	3692	1200	75.2	3660	1200	79.5
864	2875	1200	79.5	2720	1087	80.6	3684	1200	75.2	3664	1200	79.5
880	2879	1195	79.5	2684	1141	80.6	3671	1200	75.8	3675	1200	79.5
897	2860	1200	80.1	2704	1033	80.6	3700	1200	74.7	3682	1200	79.5
914	2858	1195	80.1	2709	1195	80.6	3674	1200	75.2	3685	1200	80.1
930	2874	1200	80.1	2681	1141	81.1	3694	1200	75.8	3706	1200	79.5
947	2931	1195	80.1	2706	1141	81.1	3683	1200	75.8	3706	1200	79.5
964	2933	1141	80.1	2714	1141	80.6	3681	1200	76.3	3720	1200	79.5
980	2874	1195	80.1	2709	1034	80.6	3672	1200	75.8	3713	1200	77.4
997	2931	1195	80.1	2747	1087	81.7	3713	1200	76.3	3798	1200	77.4
Min	2858			2681			3671			3660		
1 CP Max	3439			3439			4360			4360		
Min % Max	83			78			84			84		

Livermore Loops Stress Test Next or Go To Start

Livermore Loops Stress Tests - liverloopsPiA7R, liverloopsPi64 - Four Copies

The Livermore Loops Benchmark was converted to act as a stress test, following wrong numeric results being produced on an overclocked PC, using a Pentium Pro CPU. To run the reliability test, a seconds parameter is required that arranges for the initial repeat passes for each of the 72 test functions, required to produce that running time. A single log file includes results from all tests. For separate logs, copies of the program can be started from different folders. The 32 bit benchmark can be obtained from Raspberry_Pi_2_Stress_Tests.zip, and the 64 bit version in econds Log Rpi3-64-Bit-Benchmarks.tar.gz.

Whilst running, results from all tests are displayed, with the logged summary shown below. Later are processor MHz and temperature measurements for both 32 bit and 64 bit programs. The cool versions were immediately after powering on, with a room temperature around 22°C, with the hot tests following shortly afterwards. As this program has 72 different variations in code executed, temperature can go up and down, with maximum just about 80°C. For most of the time, in all cases, CPU MHz was at 1200 MHz, and this is reflected in little difference in hot and cold overall performance ratings (provided below). Here, single core performance results generally indicate faster speeds proportional to MHz speed increase.

Example Log File Entry ./liverloopsPi64 Seconds 12

Livermore Loops Benchmark armv8 64 Bit via C/C++ Fri Aug 17 12:33:02 2018

Reliability test 12 seconds each loop x 24 x 3

```
Part 1 of 3 start at Fri Aug 17 12:33:03 2018
Part 2 of 3 start at Fri Aug 17 12:38:31 2018
Part 3 of 3 start at Fri Aug 17 12:43:14 2018
```

Numeric results were as expected

MFLOPS for 24 loops

```
 530.4  296.7  513.6  450.1  198.5  191.2  629.0  424.8  450.5  229.1  145.8  208.1
 105.1  128.3  247.8  219.5  374.4  440.5  284.2  237.5  260.8   78.5  307.1  176.3
```

	Overall Ratings				
	Maximum	Average	Geomean	Harmean	Minimum
	629.0	274.7	243.8	213.4	76.8
64b Hot	622.0	272.0	241.8	212.4	77.7
64b 1 Core	720.6	320.2	285.6	251.9	94.4
32b Cool	428.4	210.4	187.2	164.4	66.0
32b Hot	386.3	209.5	187.0	164.5	66.2
32b 1 Core	462.5	243.8	215.2	185.7	65.6

	Raspbian 32 Bit				Gentoo 64 Bit			
	Cool		Hot		Cool		Hot	
Secs	MHz	°C	MHz	°C	MHz	°C	MHz	°C
0	1400	53.7	1400	58.0	1400	52.6	1399	60.1
15	1400	65.5	1199	69.8	1400	60.1	1400	69.3
31	1400	69.8	1200	70.4	1400	66.6	1200	70.4
46	1200	70.9	1200	73.1	1200	70.9	1200	74.1
62	1200	70.4	1200	72.5	1200	70.9	1200	76.8
77	1199	70.4	1200	71.4	1200	69.8	1200	74.1
93	1200	70.4	1200	72.5	1200	70.9	1200	74.1
280	1200	72.0	1200	74.1	1200	70.9	1200	74.1
296	1200	72.0	1200	74.1	1200	72.0	1200	75.8
311	1200	73.6	1200	75.2	1200	70.9	1200	75.2
327	1200	74.1	1200	76.3	1200	72.0	1200	76.3
343	1200	73.6	1200	76.3	1200	74.1	1200	77.4
358	1200	75.8	1200	77.4	1200	73.6	1200	77.4
374	1200	75.2	1200	78.4	1200	77.4	1195	79.5
389	1200	74.7	1200	76.3	1200	75.2	1195	79.5
607	1200	76.3	1200	77.4	1199	74.1	1200	76.8
623	1200	75.8	1200	77.9	1200	75.2	1200	77.9
639	1200	77.4	1200	79.0	1200	75.8	1200	78.4
654	1200	77.4	1200	79.5	1200	78.4	1200	79.5
670	1200	77.4	1200	78.4	1200	78.4	1141	80.1
685	1200	76.3	1200	78.4	1200	76.3	1199	79.5
701	1200	76.8	1200	78.4	1200	76.8	1200	79.0
763	1200	76.3	1200	78.4	1200	76.8	1200	78.4
779	1200	75.2	1200	77.9	1200	77.4	1200	78.4
794	1200	76.8	1200	78.4	1200	75.2	1200	77.9
810	1200	75.8	1200	78.4	1200	76.3	1199	77.4
826	1200	75.2	1200	77.4	1200	76.3	1200	77.9
841	1200	75.2	1200	77.4	1200	77.4	1200	77.4
857	1200	75.2	1200	76.3	1200	74.1	1200	77.4
872	1200	76.3	1200	77.9	1200	75.8	1200	76.8

OpenGL Stress Tests - videogl32, videogl64 - One Copy

These use the OpenGL GLUT Benchmarks that have command parameters for window width and height, plus running time in minutes. Default, with no size parameters, is current monitor resolution, in this case 1920 x 1080. The first exercise was to execute short runs to determine system loading of the various test functions. The commands, from different terminal windows, are shown below. Examples of the usual RPiHeatMHzVolts program and stress test outputs are provided later. For those who do not know, example vmstat results are shown. CPU utilisation is the sum of user and system entries (us+sy), where this relates to four core loading. So, 12% is the same as 48% of a single core. In this care, maximum temperatures and CPU utilisation, along with slowest FPS speed, provide the heaviest loading, using the Tiled Kitchen function.

Hot Test - The Tiled Kitchen test was then run for 15 minute test periods, when room temperature was about 22°C. As indicated, a first run indicated a constant 20 FPS speed (rounded up or down), nearly reaching 70°C, where MHz reduces to 1200. However, the short term FPS, displayed during the tests, sometimes indicated 19 FPS, indicting that the MHz can vary quite rapidly. The hot run followed almost immediately afterwards, when the display recorded between 17 and 20 FPS. The unsynchronised variations in temperatures, voltage and CPU MHz again suggest rapid variations. Recorded temperature reached 70.9°C.

Extended Power Cable Test - As mentioned with MultiThreading Benchmarks, the CPU cores can run slowly with longer than normal power supply cables. A short videogl64 test was run with a one metre extension cable, on the 2.5A power supply. This time, core voltage measurements were included, indicating 1.2 volts, instead of 1.35 (therabouts), with 600 MHz and 8 FPS. The Pi 3B+ deserves a commendation for actually running in these circumstances (a permanent way of running cool?).

```
 Commands    ./RPiHeatMHzVolts64G passes 7 seconds 10
             ./videogl64  test n, mins 1, where n = 1 to 6
             ./vmstat 10 7  - for 7 samples every 10 seconds, example output next

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st

 1  0      0 530084  22812 216736    0    0     0    15 3029 1131  9  3 89  0  0
```

Test	Raspbian 32 bit FPS	%CPUt	°C	Gentoo 64 bit FPS	%CPUt	°C
1 Few Objects	49	48	58.5	54	64	62.8
2 All Objects, No Textures	47	60	59.1	52	80	63.4
3 Few Objects, With Textures	36	80	60.1	39	96	64.5
4 All Objects, With Textures	32	100	61.2	35	96	65.5
5 WireFrame Kitchen	23	116	61.2	27	124	66.6
6 Tiled Kitchen	14	112	61.2	20	116	66.6
6 Tiled Kitchen 15 minutes	14	112	69.3	20	116	69.8

Hot Run

```
OpenGL Reliability Test 64 Bit Version 1, Mon Aug 20 17:12:38 2018

Display 1920 x 1080  Tiled Kitchen, Test for 15 minutes

Normal Output                                Part RPiHeatMHzVolts Results

Start                                       1400 ARM MHz, core volt=1.3438V, temp=62.8°C
Test 6  Tiled Kitchen, 30 seconds, 20 FPS   1400 ARM MHz, core volt=1.3500V, temp=67.7°C
Test 6  Tiled Kitchen, 30 seconds, 20 FPS   1400 ARM MHz, core volt=1.3500V, temp=69.3°C
Test 6  Tiled Kitchen, 30 seconds, 20 FPS   1400 ARM MHz, core volt=1.2375V, temp=69.8°C
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   1400 ARM MHz, core volt=1.3500V, temp=69.8°C
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   1200 ARM MHz, core volt=1.2375V, temp=69.8°C
Test 6  Tiled Kitchen, 30 seconds, 19 FPS    Continued 19 FPS next 14 entries
Test 6  Tiled Kitchen, 30 seconds, 19 FPS    5 at 1200 MHz, 3 at 1.2375V
Test 6  Tiled Kitchen, 30 seconds, 19 FPS    Temperatures 69.8°C and 70.9°C up and down

Test 6  Tiled Kitchen, 30 seconds, 18 FPS   1400 ARM MHz, core volt=1.2375V, temp=69.8°C
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   1400 ARM MHz, core volt=1.3500V, temp=70.4°C
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   Coninued 19 FPS 10 entries to end
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   2 at 1200 MHz, 5 at 1.2375V
Test 6  Tiled Kitchen, 30 seconds, 19 FPS   Temperatures 69.8°C and 70.9°C up and down

Extended Power Cable Test OpenGL Reliability Program 64 Bit Version 1

Test 6  Tiled Kitchen, 30 seconds, 8 FPS
Test 6  Tiled Kitchen, 30 seconds, 8 FPS

Temperature and CPU MHz Measurement

Using 70 samples at 1 second intervals

Seconds
   0.0     1400 scaling MHz,   1400 ARM MHz, core volt=1.2000V, temp=52.6°C
   1.0     1400 scaling MHz,    600 ARM MHz, core volt=1.2000V, temp=52.6°C
   2.4     1400 scaling MHz,    600 ARM MHz, core volt=1.2000V, temp=52.6°C
   3.8     1400 scaling MHz,   1400 ARM MHz, core volt=1.2000V, temp=52.6°C

   To end 53.2°C to 55.3°C, mainly 600 MHz, 1.2000V, some 1400 MHz, 1.3438V
```

OpenGL + Three Copies CPU Stress Tests

Next we have the 64 bit stress tests running the OpenGL program at the same time as three copies of the Integer, Floating Point or Livermore Loops Stress Tests. All reached the critical 80°C barrier, where CPU frequency reduces from 1200 MHz. Room temperature was around 22°C, with the higher floating point test starting point not maintaining the disadvantage after a short time. The details are based on approximate 30 second intervals, average overall OpenGL Frames Per Second and CPU performance, with CPU MHz and temperature averages derived from not necessarily accurate samples.

The stand alone OpenGL Tests CPU utilisation report indicated that more than one core was being used for some of the time. This probably lead to greater floating point and OpenGL performance reductions, compared with the integer tests, with similar average temperatures. The run using Livermore Loops was the least affected.

These tests were run with the main board in a simple plastic case and the CPU having no heatsink, The following section includes repeats of the tests, with the system in a FLIRC case, where the whole aluminium case becomes the heatsink, and lead to significantly lower CPU temperatures during earlier Raspberry Pi 3B stress tests - see Raspberry Pi 2 and 3 Stress Tests (possibly slow to load archived copy) and ResearchGate Pi 3B Report.

Raspberry Pi 3B+ 64 Bit Stress Tests

Secs	stressIntPi64+OGL			burninfpuPi64+OGL			liverloopsPi64+OGL		
	MHz	FPS	°C	MHz	FPS	°C	MHz	FPS	°C
0	1400		55.8	1400		66.1	1400		49.9
30	1200	16	69.8	1200	9	72.5	1400	17	67.7
60	1200	17	73.1	1200	11	75.2	1200	17	70.9
90	1200	16	75.2	1200	10	76.3	1200	16	70.9
120	1200	16	76.3	1200	10	75.8	1200	16	70.4
150	1200	16	77.4	1200	11	77.4	1200	16	70.9
180	1200	16	78.4	1200	10	77.9	1200	16	73.1
210	1200	16	79.5	1200	10	78.4	1200	15	73.6
240	1200	16	80.1	1200	11	79.5	1200	16	74.1
270	1141	16	80.6	1200	12	80.1	1200	16	74.1
300	1034	15	81.1	1195	13	80.6	1200	16	74.7
330	1141	15	80.6	1195	11	80.6	1200	16	79.0
360	1141	15	81.7	1087	11	80.6	1200	16	77.9
390	1034	15	81.1	1141	12	80.6	1200	16	77.4
420	1034	15	81.7	1141	11	81.1	1200	15	78.4
450	1034	14	80.6	1141	11	80.6	1200	15	79.5
480	1141	15	80.6	1141	10	80.6	1200	17	79.0
510	980	15	81.7	1141	11	81.1	1200	16	78.4
540	1034	15	80.6	1034	11	80.6	1200	16	79.0
570	1034	14	81.7	1087	11	80.6	1195	16	80.6
600	1034	15	81.1	1141	11	81.7	1034	17	80.6
630	1034	14	81.7	1087	11	81.1	1141	16	81.1
660	1034	14	81.1	1087	10	80.6	1195	16	80.6
690	1034	14	81.7	1034	9	81.1	1200	15	79.5
720	1034	14	81.7	1140	11	81.7	1141	16	80.6
750	980	14	81.7	1141	10	81.7	1141	16	79.5
780	1034	14	81.7	1034	10	81.1	1141	16	80.6
810	1034	14	82.7	1034	8	82.2	1141	16	80.6
840	1034	14	82.2	1141	10	81.1	1141	16	81.1
870	1034	14	82.2	1033	10	81.7	1200	16	76.8
900	1141	13	79.5	1141	10	81.1	1200	16	75.2
Average	1093	14.9	80.0	1137	10.5	79.8	1189	16.0	76.9
%Av/Max	78	75		81	53		85	80	

Performance	MB/sec	MFLOPS	MFLOPS
Average Was	2376	2880	229.7
Max	3168	4360	285.6
%	75.0	66.1	80

Max is typical average performance testing a single core

FLIRC Case Stress Tests or Go To Start

FLIRC Case Stress Tests

The FLIRC case has box like extension under the lid that presses down on the processor, via a supplied thermal pad, enabling the whole aluminium case become a heatsink. The Pi 3B+ was fitted in a new one and the 64 bit integer and floating point plus OpenGL tests repeated, essentially starting at the same room temperature as the earlier exercise. The first integer test was run shortly after powering on. The next one shown was for a third run, the second one being spoiled by accidentally allowing power saving to stop the display. The floating point sessions followed after half an hour, with 9 minutes in between.

Performance using the FLIRC case was clearly superior to that from using a plastic case with no heatsink filled on the Pi board but, with varying starting conditions, it is difficult to be precise. In the latest results, there is little sign of core temperatures reaching 70°C, with CPU MHz almost always at 1400 MHz. Average graphics Frames Per Second were quite close to the maximum possible, with all cores fully utilised (at 19 FPS), 27% to 75% faster than using the plastic case. Also average MB/second, from the Integer tests, was 24% to 27% faster. and floating point MFLOPS providing 42% to 44% improvement.

Raspberry Pi 3B+ 64 Bit Stress Tests

	stressIntPi64+OGL			stressIntPi64+OGL			burninfpuPi64+OGL			burninfpuPi64+OGL		
Secs	**MHz**	**FPS**	**°C**	**MHz**	**FPS**	**°C**	**MHz**	**FPS**	**°C**	**MHz**	**FPS**	**°C**
0	1400		37.6	1400		53.7	1400		46.2	1400		50.5
30	1400	18	47.2	1400	19	60.1	1400	19	54.2	1400	19	56.4
60	1400	19	49.9	1400	19	63.9	1400	19	56.4	1400	19	60.1
90	1400	19	51.5	1400	19	64.5	1400	19	58.5	1400	19	61.8
120	1400	19	52.1	1400	19	66.6	1400	19	58.5	1400	18	62.3
150	1399	19	53.7	1400	19	66.6	1400	19	60.1	1400	19	63.4
180	1400	19	54.2	1400	18	68.2	1399	18	60.7	1400	18	63.4
210	1400	19	55.8	1400	19	67.7	1400	19	61.8	1400	19	64.5
240	1400	19	55.8	1400	18	68.8	1400	18	62.3	1400	19	65.0
270	1399	19	56.9	1399	19	68.8	1400	19	62.3	1400	19	65.5
300	1400	19	58.0	1400	19	69.3	1400	19	62.8	1400	19	66.1
330	1400	18	58.0	1400	19	69.3	1400	19	62.8	1400	19	66.6
360	1400	19	59.1	1400	19	69.3	1400	19	63.4	1400	18	66.6
390	1400	19	59.1	1400	18	69.8	1400	18	64.5	1400	19	67.1
420	1400	19	60.1	1200	18	69.3	1400	18	65.0	1400	19	67.7
450	1400	19	60.7	1400	18	70.4	1400	18	65.0	1400	19	67.7
480	1400	19	61.2	1400	19	69.8	1400	19	65.5	1400	19	67.7
510	1400	19	60.7	1400	19	69.8	1400	19	65.5	1400	18	68.8
540	1400	19	61.2	1400	19	69.8	1400	19	66.6	1400	19	68.8
570	1400	19	61.2	1400	19	69.8	1400	19	66.6	1400	19	69.3
600	1400	19	62.3	1400	18	69.8	1400	18	66.6	1400	19	68.8
630	1400	19	62.3	1200	18	70.4	1400	18	67.1	1400	18	68.8
660	1400	19	63.4	1400	18	70.4	1400	18	67.7	1400	19	69.3
690	1400	19	64.5	1399	18	69.8	1400	18	68.2	1400	19	69.3
720	1400	19	64.5	1400	18	69.8	1400	18	67.7	1400	19	69.3
750	1400	19	65.0	1400	18	70.4	1400	18	67.7	1400	19	69.8
780	1400	19	65.5	1400	18	69.8	1400	18	67.7	1400	19	69.3
810	1400	19	65.5	1400	18	70.4	1400	18	68.8	1200	19	69.8
840	1400	19	66.6	1400	18	69.8	1400	18	68.8	1400	18	69.8
870	1400	19	66.1	1400	18	69.8	1400	18	69.3	1400	19	69.8
900	1400	19	67.1	1400	17	70.4	1400	17	69.3	1400	19	69.8
Average	1400	18.9	59.6	1387	18.4	68.8	1400	18.4	64.4	1393	18.8	66.8
%Av/Max	100	95		99	92		100	92		99	94	

Performance	MB/sec	MB/sec	MFLOPS	MFLOPS
Average Was	3025	2954	4162	4096
Max	3168	3168	4360	4360
%	95.5	93.3	95.5	93.9

Max is typical average performance testing a single core

Assembly Code Next or Go To Start

Assembly Code

As shown in the code below, I have disassembled my MemSpeed type benchmarks. There are calculations using intrinsic functions and normal four way unrolled C code. each of the four Vector Multiply Accumulate intrinsic statements should lead to execution of four multiplies and four adds (total of 16 floating point operations). The C code loop has four multiples and four adds, but the compilers might be expected to unroll this further, where appropriate (they didn't - is there a parameter to force this?). This lead to the fastest speeds being produced by intrinsics, using assembly code instructions shown below.

NeonSpeed - At 32 bit working the Vector Multiply Accumulate intrinsic were directly converted to NEON vmla.f32 instructions using quad word registers. The 64 bit compiler converted the intrinsics to A64 instructions "Floating-point fused multiply-add to accumulator" , using 128 bit vector registers. Next are instructions generated for normal C code, using neon and funsafe compiler directives at 32 bits and standard parameters at 64 bits, acting on single precision calculations. At 32 bits, a single SIMD NEON instruction is used - vfma.32 (Vector Fused Multiply Accumulate) with four calculations. At 64 bits, vfma is generated again. The difference in speed is apparent from using a single SIMD instruction in the loop, compared with four with intrinsics.

MemSpeed - Single Precision vs Double Precision - For the 32 bit version, using the NEON compiling parameter shown, NEON instructions were not generated, four scalar Floating-point multiply-accumulate (fmacs or fmacd) were produced instead, producing the slowest speeds. Adding that funsafe parameter produced the same vfma.f32 NEON instruction as NeonSpeed for four single precision calculations. But four vfma.f64 were generated for double precision. Yes these are NEON instructions but SISD (Single Instruction Single Data), each with data in 64 bit scalar registers.

64 Bit MemSpeed - For the four sets of calculations, the fmla vector instructions were again produced, requiring two for double precision and speed closer to that from single precision calculations.

Program Code

NEON Intrinsics

```
{
   x41 = vld1q_f32(ptrx1);
   x42 = vld1q_f32(ptrx2);
   x43 = vld1q_f32(ptrx3);
   x44 = vld1q_f32(ptrx4);
   y41 = vld1q_f32(ptry1);
   y42 = vld1q_f32(ptry2);
   y43 = vld1q_f32(ptry3);
   y44 = vld1q_f32(ptry4);
   z41 = vmlaq_f32(x41, y41, c4);
   z42 = vmlaq_f32(x42, y42, c4);
   z43 = vmlaq_f32(x43, y43, c4);
   z44 = vmlaq_f32(x44, y44, c4);
   vst1q_f32(ptrx1, z41);
   vst1q_f32(ptrx2, z42);
   vst1q_f32(ptrx3, z43);
   vst1q_f32(ptrx4, z44);
   ptrx1 = ptrx1 + 16;
   ptry1 = ptry1 + 16;
   ptrx2 = ptrx2 + 16;
   ptry2 = ptry2 + 16;
   ptrx3 = ptrx3 + 16;
   ptry3 = ptry3 + 16;
   ptrx4 = ptrx4 + 16;
   ptry4 = ptry4 + 16;
}
```

MemSpeed and NEONSpeed C Code for Compilation Single and Double Precision

```
for (m=0; m<kd; m=m+inc)
{
  xn[m]   = xn[m]   + sumn * yn[m];
  xn[m+1] = xn[m+1] + sumn * yn[m+1];
  xn[m+2] = xn[m+2] + sumn * yn[m+2];
  xn[m+3] = xn[m+3] + sumn * yn[m+3];            }
(
```

```
 ######################################################################
```

NEON Speed Intrinsics

32 Bit

1173 MFLOPS

```
.L75:
   add     r0, r3, #48
   add     ip, r3, #32
   add     lr, r3, #16
   add     r10, r2, #48
   add     r7, r2, #32
   add     r4, r2, #16
   vld1.32 {d24-d25}, [r3]
   vld1.32 {d18-d19}, [r0]
   vld1.32 {d20-d21}, [ip]
   vld1.32 {d22-d23}, [lr]
   vld1.32 {d26-d27}, [r2]
   vld1.32 {d6-d7}, [r10]
   vld1.32 {d30-d31}, [r7]
   vld1.32 {d28-d29}, [r4]
   vmla.f32        q9, q3, q8
   vmla.f32        q10, q15, q8
   vmla.f32        q11, q14, q8
   vmla.f32        q12, q13, q8
   add     r1, r1, #1
   add     r2, r2, #64
   cmp     r1, r5
   vst1.32 {d24-d25}, [r3]
   vst1.32 {d22-d23}, [lr]
   add     r3, r3, #64
   vst1.32 {d20-d21}, [ip]
   vst1.32 {d18-d19}, [r0]
   bne     .L75
```

64 Bit

1277 MFLOPS

```
.L13:
   ldr     q4, [x3, -16]
   add     x3, x3, 64
   ldr     q3, [x3, -64]
   add     x1, x1, 64
   ldr     q2, [x3, -48]
   ldr     q1, [x3, -32]
   cmp     x3, x2
   ldr     q16, [x1, -64]
   ldr     q7, [x1, -48]
   ldr     q6, [x1, -32]
   ldr     q5, [x1, -16]
   fmla    v4.4s, v0.4s, v16.4s
   fmla    v3.4s, v0.4s, v7.4s
   fmla    v2.4s, v0.4s, v6.4s
   fmla    v1.4s, v0.4s, v5.4s
   str     q4, [x3, -80]
   str     q3, [x3, -64]
   str     q2, [x3, -48]
   str     q1, [x3, -32]
   bne     .L13
```

More Below or Go To Start

##

NEON Speed Normal

-mfpu=neon-vfpv4
-funsafe-math-optimizations

797 MFLOPS

32 Bit

```
.L54:
        vld1.32 {q9}, [r2]
        vld1.32 {q8}, [r3]
        add     r1, r1, #1
        add     r2, r2, #16
        cmp     r1, r4
        vfma.f32        q8, q9, q7
        vst1.32 {q8}, [r3]
        add     r3, r3, #16
        bcc     .L54
```

-march=armv8-a

681 MFLOPS

64 bit

```
.L37:
        ldr     q0, [x0, x26]
        add     w1, w1, 1
        ldr     q1, [x0, x28]
        cmp     w24, w1
        fmla    v0.4s, v1.4s, v2.4s
        str     q0, [x0, x26]
        add     x0, x0, 16
        bhi     .L37
```

##

MemSpeed 32 Bit Single and Double Precision

Parameters -mfpu=neon-vfpv4

Single Precision

MFLOPS 532

```
.L45:
   mov     ip, r2
   flds    s15, [r3]
   flds    s11, [ip]
   flds    s12, [r3, #-12]
   flds    s13, [r3, #-8]
   flds    s14, [r3, #-4]
   flds    s8, [r2, #-12]
   flds    s9, [r2, #-8]
   flds    s10, [r2, #-4]
   fmacs   s15, s11, s30
   fmacs   s12, s8, s30
   fmacs   s13, s9, s30
   fmacs   s14, s10, s30
   add     r2, r2, #16
   fmrs    ip, s15
   fsts    s12, [r3, #-12]
   fsts    s13, [r3, #-8]
   fsts    s14, [r3, #-4]
   str     ip, [r3], #16   @ float
   cmp     r3, r6
   bne     .L45
```

Double Precision

mFLOPS 238

```
.L31:
   fldd    d5, [r2, #-24]
   fldd    d6, [r3, #-24]
   fldd    d7, [r3, #-16]
   fldd    d4, [r3, #-8]
   mov     r6, r2
   fmacd   d6, d5, d8
   fldd    d3, [r3]
   add     r2, r2, #32
   fstd    d6, [r3, #-24]
   fldd    d6, [r2, #-48]
   fmacd   d7, d6, d8
   fstd    d7, [r3, #-16]
   fldd    d7, [r2, #-40]
   fmacd   d4, d7, d8
   fstd    d4, [r3, #-8]
   fldd    d7, [r6]
   fmacd   d3, d7, d8
   fmrrd   r8, r9, d3
   strd    r8, [r3], #32
   cmp     r3, r1
   bne     .L31
```

##

More MemSpeed 32 Bit Single and Double Precision

Parameters -mfpu=neon-vfpv4 -funsafe-math-optimizations

Single Precision

MFLOPS 695

```
.L44:
   vld1.64 {d16-d17}, [r3:64]
   vld1.64 {d18-d19}, [r1:64]
   add     r2, r2, #1
   add     r1, r1, #16
   cmp     r4, r2
   add     r3, r3, #16
   vfma.f32        q8, q9, q7
   vstr    d16, [r3, #-16]
   vstr    d17, [r3, #-8]
   bhi     .L44
```

Double Precision

MFLOPS 236 MLOPS

```
.L28:
   fldd    d17, [r2, #-24]
   fldd    d16, [r3, #-24]
   fldd    d18, [r3, #-16]
   vfma.f64        d16, d17, d8
   mov     r4, r2
   fldd    d17, [r3, #-8]
   add     r2, r2, #32
   fcpyd   d19, d16
   fldd    d16, [r3]
   fstd    d19, [r3, #-24]
   fldd    d19, [r2, #-48]
   vfma.f64        d18, d19, d8
   fstd    d18, [r3, #-16]
   fldd    d18, [r2, #-40]
   vfma.f64        d17, d18, d8
   fstd    d17, [r3, #-8]
   fldd    d17, [r4]
   vfma.f64        d16, d17, d8
   fmrrd   r4, r5, d16
   strd    r4, [r3], #32
   cmp     r3, r1
   bne     .L28
```

More Below or Go To Start

```
#########################################################################
```

MemSpeed 64 Bit Single and Double Precision

```
   Parameters - -march=armv8-a
   Single Precision                    Double Precision
   MFLOPS 726                          MFLOPS 602

.L56:                                .L34:
   ldr     q0, [x27, x0]               ldr     q5, [x2, 16]
   add     w1, w1, 1                   add     w1, w1, 1
   ldr     q1, [x23, x0]               ldr     q1, [x0, 16]
   cmp     w21, w1                     cmp     w28, w1
   fmla    v0.4s, v1.4s, v2.4s         ldr     q3, [x2], 32
   str     q0, [x27, x0]               add     x0, x0, 32
   add     x0, x0, 16                  ldr     q0, [x0, -32]
   bhi     .L56                        fmla    v1.2d, v5.2d, v2.2d
                                       fmla    v0.2d, v3.2d, v2.2d
                                       str     q1, [x0, -16]
                                       str     q0, [x0, -32]
                                       bhi     .L34
```

MP-MFLOPS on Raspberry Pi 3B+

The 32 bit compilations uses 12 scalar add and multiply instructions and 10 using fused multiply accumulate NEON type, but limited to scalar operation (SISD - Single Instructions Single Data). All the others use NEON or 64 bit vector SIMD instructions (Multiple Data), carrying out four calculations simultaneously at single precision, with 128 operations in the execution loops, or half these at double precision. Each has its own variation of fused multiply and add or subtract instructions.

In the original single precision benchmarks, the NEON version produced significantly faster performance, where the compiler converted the 32 intrinsic calculating functions into 22 instructions, with those fused operations, and a total in-loop count of 27. Performance of the first 64 bit version was degraded through making use of only 12 vector registers, for a programming function involving 23 variables, necessitating frequent load instructions. The gcc 7 compiler made use of 25 vector registers with out of loop loads to achieve similar performance as the hand code NEON benchmark. Both the 64 bit double precision benchmarks included the higher efficient code, with external data loading, but best speed was, as expected, half that for single precision SIMD calculations.

Function triadplus2

```
for(i=0; i<n; i++)
x[i] = (x[i]+a)*b-(x[i]+c)*d+(x[i]+e)*f-(x[i]+g)*h+(x[i]+j)*k
-(x[i]+l)*m+(x[i]+o)*p-(x[i]+q)*r+(x[i]+s)*t-(x[i]+u)*v+(x[i]+w)*y;
```

```
#########################################################################
```

gcc 4.9 32 bit

SP MFLOPS
797 1t 3134 4T

```
.L21:
   flds      s23, [r3]
   fadds     s15, s8, s23
   fadds     s24, s10, s23
   fadds     s31, s6, s23
   fadds     s30, s4, s23
   fnmuls    s15, s15, s7
   fadds     s29, s3, s23
   fadds     s28, s1, s23
   fadds     s27, s0, s23
   vfma.f32  s15, s9, s24
   fadds     s26, s17, s23
   fadds     s25, s18, s23
   fadds     s24, s20, s23
   fadds     s23, s21, s23
   vfma.f32  s15, s5, s31
   vfma.f32  s15, s14, s30
   vfma.f32  s15, s2, s29
   vfma.f32  s15, s13, s28
   vfma.f32  s15, s16, s27
   vfma.f32  s15, s12, s26
   vfma.f32  s15, s19, s25
   vfma.f32  s15, s11, s24
   vfma.f32  s15, s22, s23
   fstmias   r3!, {s15}
   cmp       r3, r2
   bne       .L9
```

DP MFLOPS
798 1T 3119 4T

```
.L21:
   fldd      d17, [r1]
   faddd     d16, d17, d2
   faddd     d18, d17, d0
   faddd     d25, d17, d4
   faddd     d24, d17, d6
   fnmuld    d16, d3, d16
   faddd     d23, d17, d15
   faddd     d22, d17, d13
   faddd     d21, d17, d11
   faddd     d20, d17, d9
   faddd     d19, d17, d31
   vfma.f64  d16, d18, d1
   faddd     d18, d17, d29
   faddd     d17, d17, d27
   vfma.f64  d16, d25, d5
   vfms.f64  d16, d24, d7
   vfma.f64  d16, d23, d14
   vfms.f64  d16, d22, d12
   vfma.f64  d16, d21, d10
   vfms.f64  d16, d20, d8
   vfma.f64  d16, d19, d30
   vfms.f64  d16, d18, d26
   vfma.f64  d16, d17, d28
   fstmiad   r1!, {d16}
   cmp       r1, r0
   bne       .L21
```

More Below or Go To Start

```
#########################################################################
```

gcc 6 64 bit

SP MFLOPS

1793 1T to 6981 4T

```
.L65:
   ldr     q16, [x2, x5]
   add     w6, w6, 1
   ldr     q15, [sp, 64]
   cmp     w3, w6
   ldr     q17, [sp, 80]
   ldr     q0, [sp, 112]
   fadd    v15.4s, v16.4s, v15.4s
   fmul    v15.4s, v15.4s, v17.4s
   ldr     q17, [sp, 96]
   fadd    v17.4s, v16.4s, v17.4s
   fmls    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 128]
   fadd    v17.4s, v16.4s, v0.4s
   ldr     q0, [sp, 144]
   fmla    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 160]
   fadd    v17.4s, v16.4s, v0.4s
   ldr     q0, [sp, 176]
   fmls    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 192]
   fadd    v17.4s, v16.4s, v0.4s
   ldr     q0, [sp, 208]
   fmla    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 224]
   fadd    v17.4s, v16.4s, v0.4s
   ldr     q0, [sp, 240]
   fmls    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 256]
   fadd    v17.4s, v16.4s, v0.4s
   ldr     q0, [sp, 272]
   fmla    v15.4s, v17.4s, v0.4s
   ldr     q0, [sp, 288]
   fadd    v17.4s, v16.4s, v0.4s
   fmls    v15.4s, v17.4s, v14.4s
   fadd    v17.4s, v16.4s, v13.4s
   fmla    v15.4s, v17.4s, v12.4s
   fadd    v17.4s, v16.4s, v11.4s
   fadd    v16.4s, v16.4s, v9.4s
   fmls    v15.4s, v17.4s, v10.4s
   fmla    v15.4s, v16.4s, v8.4s
   str     q15, [x2, x5]
   add     x5, x5, 16
   bhi     .L65
```

```
#########################################################################
```

gcc 7

SP MFLOPS

2800 1T to 10608 4T

```
.L51:
   ldr     q15, [x2, x3]
   add     w4, w4, 1
   cmp     w4, w6
   fadd    v0.4s, v15.4s, v14.4s
   fadd    v17.4s, v15.4s, v12.4s
   fmul    v0.4s, v0.4s, v13.4s
   fmls    v0.4s, v17.4s, v11.4s
   fadd    v17.4s, v15.4s, v10.4s
   fmla    v0.4s, v17.4s, v9.4s
   fadd    v17.4s, v15.4s, v8.4s
   fmls    v0.4s, v17.4s, v31.4s
   fadd    v17.4s, v15.4s, v30.4s
   fmla    v0.4s, v17.4s, v29.4s
   fadd    v17.4s, v15.4s, v16.4s
   fmls    v0.4s, v17.4s, v28.4s
   fadd    v17.4s, v15.4s, v27.4s
   fmla    v0.4s, v17.4s, v26.4s
   fadd    v17.4s, v15.4s, v25.4s
   fmls    v0.4s, v17.4s, v24.4s
   fadd    v17.4s, v15.4s, v23.4s
   fmla    v0.4s, v17.4s, v22.4s
   fadd    v17.4s, v15.4s, v21.4s
   fadd    v15.4s, v15.4s, v19.4s
   fmls    v0.4s, v17.4s, v20.4s
   fmla    v0.4s, v15.4s, v18.4s
   str     q0, [x2, x3]
   add     x3, x3, 16
   bcc     .L51
```

DP MFLOPS

1405 1T to 4398 4T

```
.L84:
   ldr     q16, [x2, x0]
   add     w3, w3, 1
   cmp     w3, w6
   fadd    v15.2d, v16.2d, v14.2d
   fadd    v17.2d, v16.2d, v12.2d
   fmul    v15.2d, v15.2d, v13.2d
   fmls    v15.2d, v17.2d, v11.2d
   fadd    v17.2d, v16.2d, v10.2d
   fmla    v15.2d, v17.2d, v9.2d
   fadd    v17.2d, v16.2d, v8.2d
   fmls    v15.2d, v17.2d, v31.2d
   fadd    v17.2d, v16.2d, v30.2d
   fmla    v15.2d, v17.2d, v29.2d
   fadd    v17.2d, v16.2d, v28.2d
   fmls    v15.2d, v17.2d, v0.2d
   fadd    v17.2d, v16.2d, v27.2d
   fmla    v15.2d, v17.2d, v26.2d
   fadd    v17.2d, v16.2d, v25.2d
   fmls    v15.2d, v17.2d, v24.2d
   fadd    v17.2d, v16.2d, v23.2d
   fmla    v15.2d, v17.2d, v22.2d
   fadd    v17.2d, v16.2d, v21.2d
   fadd    v16.2d, v16.2d, v19.2d
   fmls    v15.2d, v17.2d, v20.2d
   fmla    v15.2d, v16.2d, v18.2d
   str     q15, [x2, x0]
   add     x0, x0, 16
   bcc     .L84
```

DP MFLOPS

1403 1T 4492 4T

```
.L44:
   ldr     q15, [x3, x2]
   add     w4, w4, 1
   cmp     w4, w5
   fadd    v7.2d, v15.2d, v14.2d
   fadd    v16.2d, v15.2d, v12.2d
   fmul    v7.2d, v7.2d, v13.2d
   fmls    v7.2d, v16.2d, v11.2d
   fadd    v16.2d, v15.2d, v10.2d
   fmla    v7.2d, v16.2d, v9.2d
   fadd    v16.2d, v15.2d, v8.2d
   fmls    v7.2d, v16.2d, v31.2d
   fadd    v16.2d, v15.2d, v30.2d
   fmla    v7.2d, v16.2d, v29.2d
   fadd    v16.2d, v15.2d, v28.2d
   fmls    v7.2d, v16.2d, v27.2d
   fadd    v16.2d, v15.2d, v26.2d
   fmla    v7.2d, v16.2d, v25.2d
   fadd    v16.2d, v15.2d, v24.2d
   fmls    v7.2d, v16.2d, v23.2d
   fadd    v16.2d, v15.2d, v22.2d
   fmla    v7.2d, v16.2d, v21.2d
   fadd    v16.2d, v15.2d, v20.2d
   fadd    v15.2d, v15.2d, v18.2d
   fmls    v7.2d, v16.2d, v19.2d
   fmla    v7.2d, v15.2d, v17.2d
   str     q7, [x3, x2]
   add     x2, x2, 16
   bcc     .L44
```

More Below or Go To Start

```
#########################################################################
```

gcc6 neon
SP MFLOPS
2999 1T to 11563 4T

```
.L41:
   ldr     q1, [x1]
   ldr     q0, [sp, 64]
   fadd    v18.4s, v20.4s, v1.4s
   fadd    v17.4s, v22.4s, v1.4s
   fadd    v0.4s, v0.4s, v1.4s
   fadd    v16.4s, v24.4s, v1.4s
   fadd    v7.4s, v26.4s, v1.4s
   fadd    v6.4s, v28.4s, v1.4s
   fadd    v5.4s, v30.4s, v1.4s
   fmul    v0.4s, v0.4s, v19.4s
   fadd    v4.4s, v10.4s, v1.4s
   fadd    v3.4s, v12.4s, v1.4s
   fadd    v2.4s, v14.4s, v1.4s
   fadd    v1.4s, v8.4s, v1.4s
   fmls    v0.4s, v21.4s, v18.4s
   fmla    v0.4s, v23.4s, v17.4s
   fmls    v0.4s, v25.4s, v16.4s
   fmla    v0.4s, v27.4s, v7.4s
   fmls    v0.4s, v29.4s, v6.4s
   fmla    v0.4s, v31.4s, v5.4s
   fmls    v0.4s, v9.4s, v1.4s
   fmla    v0.4s, v4.4s, v11.4s
   fmls    v0.4s, v3.4s, v13.4s
   fmla    v0.4s, v2.4s, v15.4s
   str     q0, [x1], 16
   cmp     x1, x0
   bne     .L41
```

C code
for(i=0; i<n; i=i+4)

```
{
  x41 = vld1q_f32(ptrx1);
  z41 = vaddq_f32(x41, a41);
  z41 = vmulq_f32(z41, b41);
  z42 = vaddq_f32(x41, c41);
  z42 = vmulq_f32(z42, d41);
  z41 = vsubq_f32(z41, z42);
  z42 = vaddq_f32(x41, e41);
  z42 = vmulq_f32(z42, f41);
  z41 = vaddq_f32(z41, z42);
  z42 = vaddq_f32(x41, g41);
  z42 = vmulq_f32(z42, h41);
  z41 = vsubq_f32(z41, z42);
  z42 = vaddq_f32(x41, j41);
  z42 = vmulq_f32(z42, k41);
  z41 = vaddq_f32(z41, z42);
  z42 = vaddq_f32(x41, l41);
  z42 = vmulq_f32(z42, m41);
  z41 = vsubq_f32(z41, z42);
  z42 = vaddq_f32(x41, o41);
  z42 = vmulq_f32(z42, p41);
  z41 = vaddq_f32(z41, z42);
  z42 = vaddq_f32(x41, q41);
  z42 = vmulq_f32(z42, r41);
  z41 = vsubq_f32(z41, z42);
  z42 = vaddq_f32(x41, s41);
  z42 = vmulq_f32(z42, t41);
  z41 = vaddq_f32(z41, z42);
  z42 = vaddq_f32(x41, u41);
  z42 = vmulq_f32(z42, v41);
  z41 = vsubq_f32(z41, z42);
  z42 = vaddq_f32(x41, w41);
  z42 = vmulq_f32(z42, y41);
  z41 = vaddq_f32(z41, z42);
  vst1q_f32(ptrx1, z41);
  ptrx1 = ptrx1 + 4;
}
```

System ID Next or Go To Start

System ID

The benchmarks obtain information on CPU hardware characteristics and version of Linux from files /proc/cpuinfo and /proc/version. Below are details provided from 32 bit Raspbian and 64 bit Gentoo and they are the same for both Raspberry Pi 3B and 3B+. Raspbian BogoMIPS appears to depend on the CPU MHz frequency governor, with the default "On-Demand" setting, 38.4 was indicated but 89.6 with the "Performance" option or 76.8 using an earlier version of Raspbian.

	32 bit Raspbian	**64 Bit Gentoo**
processor	0 to 3	0 to 3
model name	ARMv7 Processor rev 4 (v7l)	
BogoMIPS	38.40 or 89.6	38.40
Features	half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm crc32	fp asimd evtstrm crc32 cpuid
CPU implementer	0x41	0x41
CPU architecture:	7	8
CPU variant	0x0	0x0
CPU part	0xd03	0xd03
CPU revision	4	4
Linux version	4.14.34-v7+ (dc4@dc4-XPS13-9333) (gcc version 4.9.3 (crosstool-NG crosstool-ng-1.22.0-88-g8460611)) #1110 SMP Mon Apr 16 15:18:51 BST 2018	4.14.31-v8-b36f4e9e1984+ (sakaki@chiyo) (gcc version 6.4.0 (Gentoo 6.4.0-r1 p1.3)) #1 SMP PREEMPT Sun Apr 1 14:15:34 BST 2018

Go To Start