PA1 ANALYSIS REPORT

Oluwafemi Ajeigbe

09-29-2022

EE 524 ADVANCED COMPUTER ARCHITECTURE

INTRODUCTION & OBJECTIVES

A benchmark test is designed to mimic a particular type of workload on a component or system to assess a program's performance (or a set of programs). The benchmarks' times depend on software, hardware, or the computer's architecture. Benchmarking can be used to measure differing performances across different systems. A code is usually tested on different computer architectures to see how it performs. In the case of this assignment, we were asked to measure how the performance would scale up with parallel threads used in a multi-core system. I carried out this test on four benchmark programs; fluidanimate, blackscholes, basicmaths, and qsort, and below are some of the data points generated after the test.

EVALUATION

ENVIRONMENTAL SETUP

The benchmark was run on an ODROID board with Linux using ***perf***. Perf is a profiler tool for Linux-based systems that abstracts CPU hardware differences in Linux performance measurements and presents a simple command-line interface. The output for each benchmark run was stored on text files; I used the SCP command to move the text from the board to the sig server and then to my PC. A Python script was used to read the lines of this text data onto a data frame. I then exported this data to an excel sheet.

RESULTS AND DISCUSSIONS

Figures 1,2,3,4,5 below show the average performance counters per 1000 instructions. These are plots that show the performance of each benchmark on the following metrics.

- The average execution times,
- The average branch misprediction rate
- and The L1 and L2 cache refill rates.

In Figure 1, we notice that the fluidanimate benchmark had the longest execution times per 1000 instructions when measured against varying core numbers. Figure 2 shows that the basicmaths benchmark had the highest branch misprediction rates per 1000 instructions when measured against varying core numbers. Figure 3 posits that blackscholes had about the highest L1 data cache refill rate per 1000 instructions when measured against varying core numbers. Figure 4 suggests that the blackscholes benchmark also had about the highest L1 instruction cache refill rate per 1000 instructions when measured against varying core numbers, which helped scale down the execution time as compared to the fluidanimate benchmark. Finally, figure 5 shows that the fluidanimate benchmark has the highest L2 data cache refill rate.

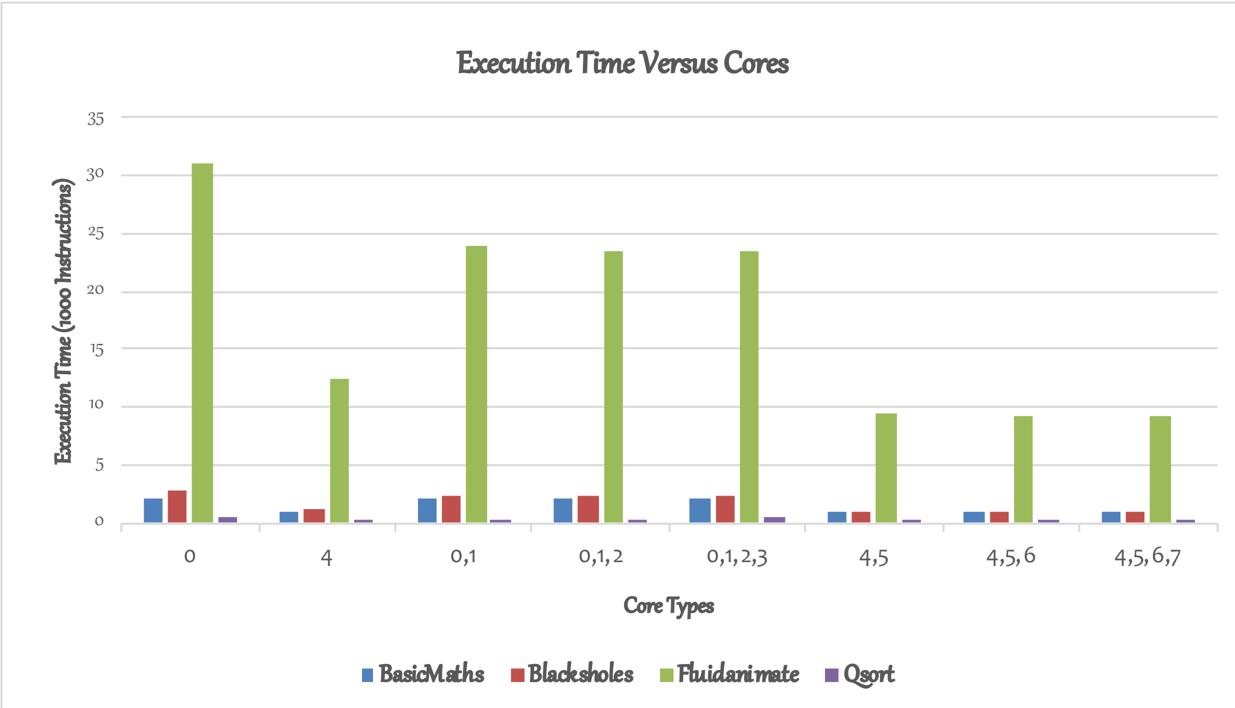

Figure 1: Average Execution time Per Cores

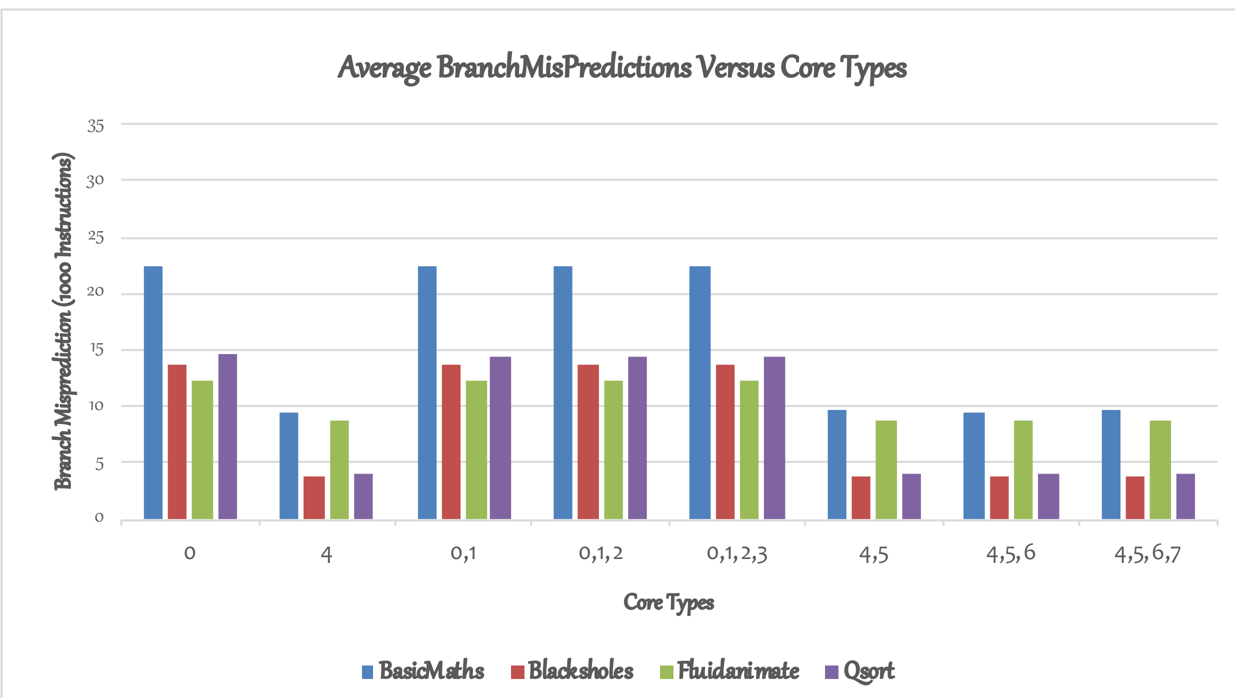

Figure 2: Average BranchMisPredictions Versus Core Types

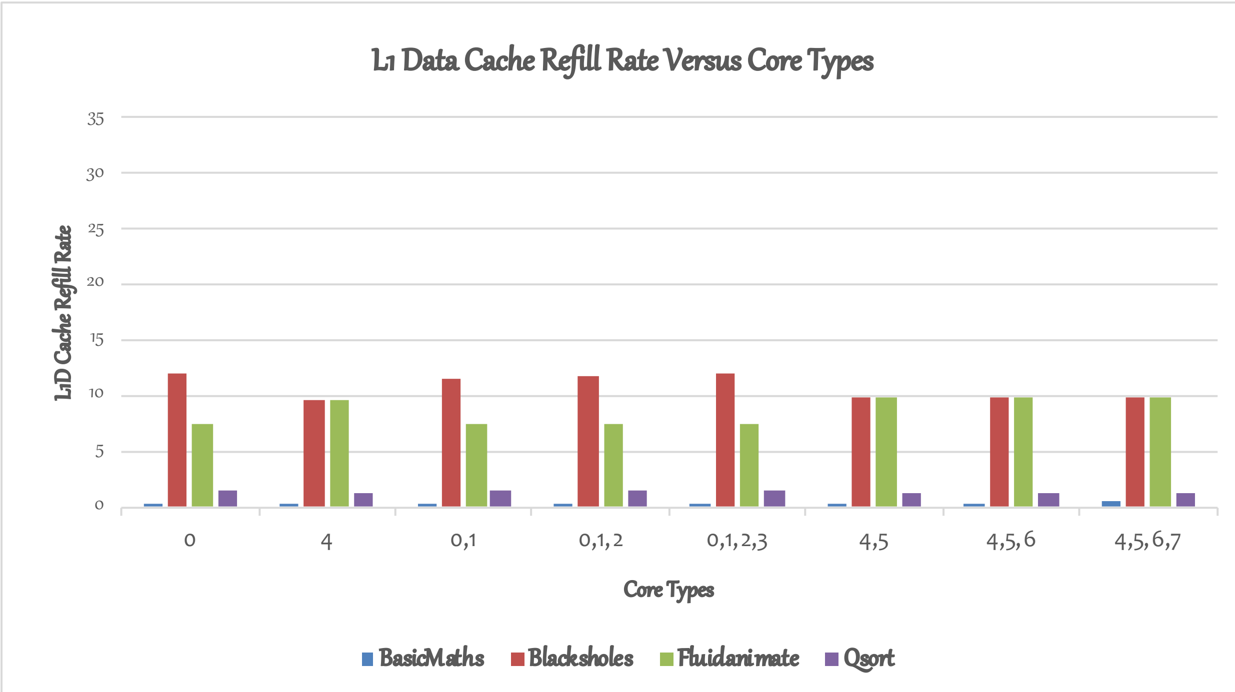

Figure 3: L1 Data Cache Refill Versus Core Types

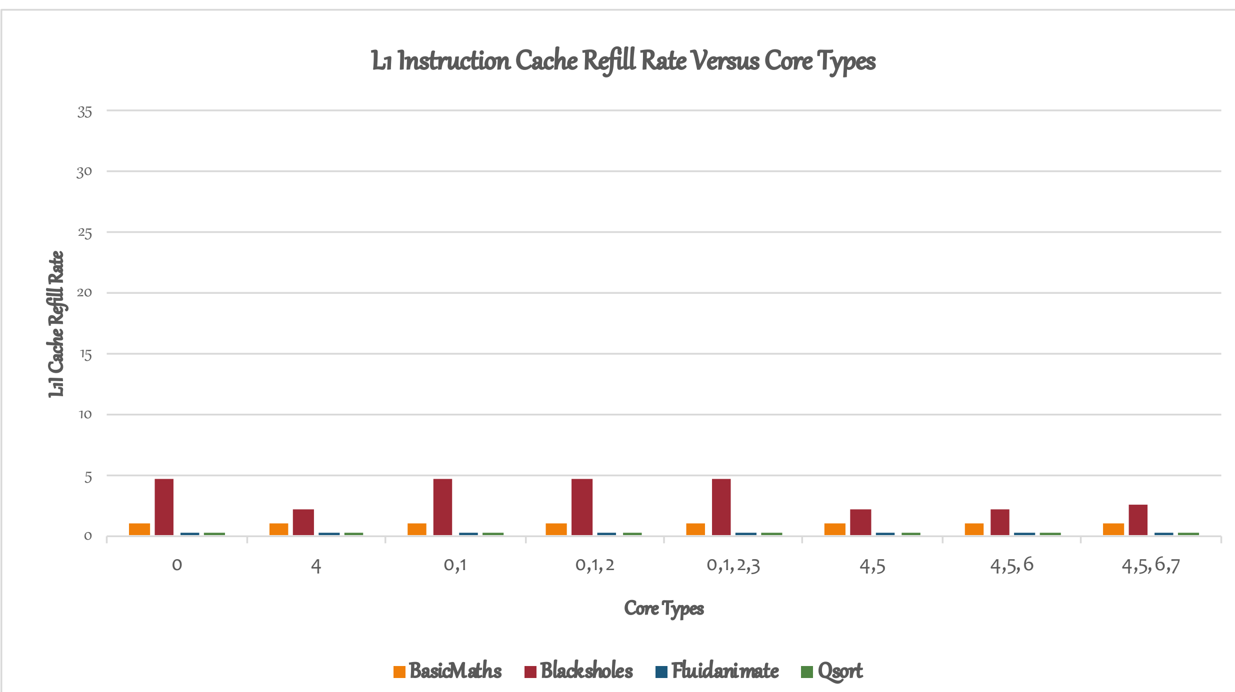

Figure 4: L1 Instruction Cache Refill Versus Core Types

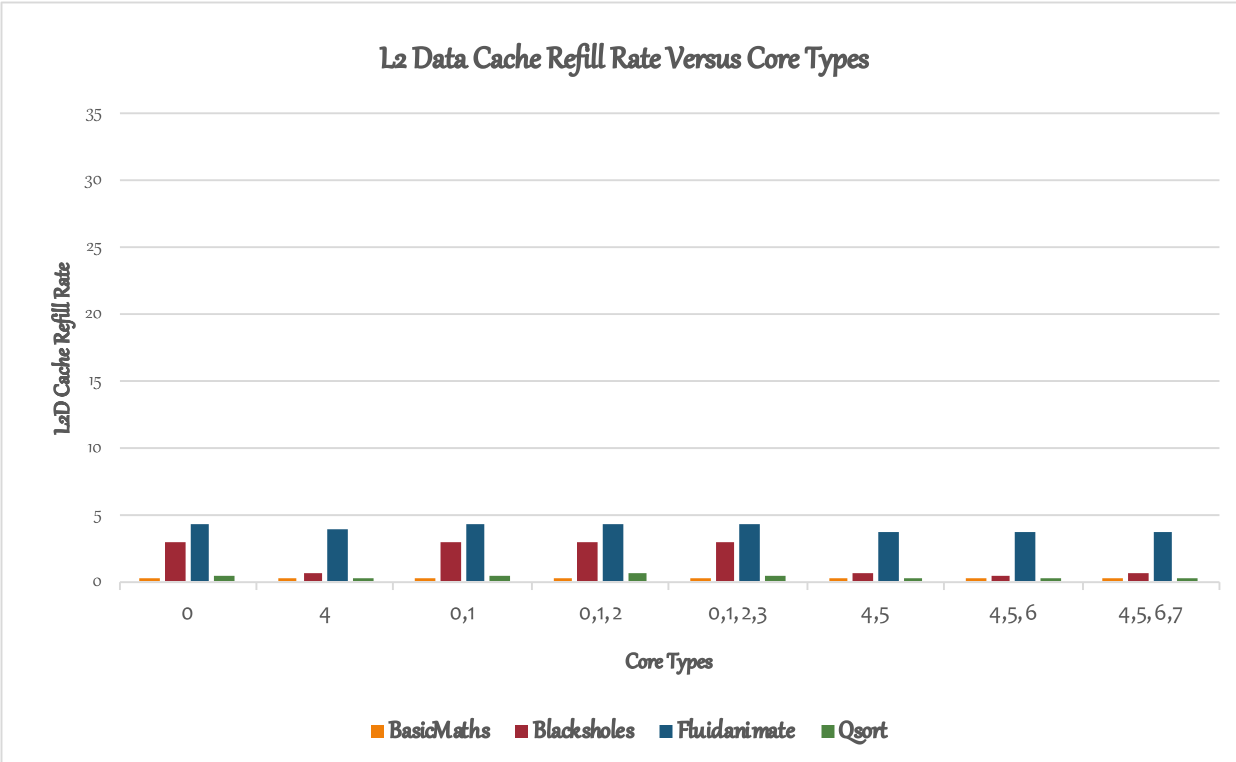

Figure 5: L2 Data Cache Refill Versus Core Types

BENCHMARK RESULTS:

1. FLUIDANIMATE: With the fluidanimate benchmark test, there were a couple of observations:

    a. The throughput was way better when we tested the program against four cores and four threads. It seems these patterns occurred with both core types. This posits that the higher the cores-threads, the faster the execution time. For example, ARM 15: 4 threads and 4,5,6,7 (4 cores) used about 2 seconds to execute the exact instructions "4 threads and 4 (1 core)" executed in 6 seconds. The trend observed from the data points suggests that the more cores and threads, the faster the execution time. This means the system could perform parallel instruction execution (because of the threads) on multiple cores (increased core number). This pattern was consistent with both core types; ARM7 (0,1,2,3) and ARM15(4,5,6,7).

    b. The branch misprediction rate was a little different for both core types. With ARM7= 12 and ARM15= 8. Better branch prediction means less time wasted speculatively executing instructions that never actually need to be executed. This was why the ARM15 cores (4 threads 4 cores) performed better (1.9secs) in terms of execution times than ARM7 (4 threads 4 cores) (4secs).

    c. L1 and L2 cache refill rates were also almost constant across the two core types.

    d. As compared to blackscholes, the execution times was way higher even with about the same instruction size.

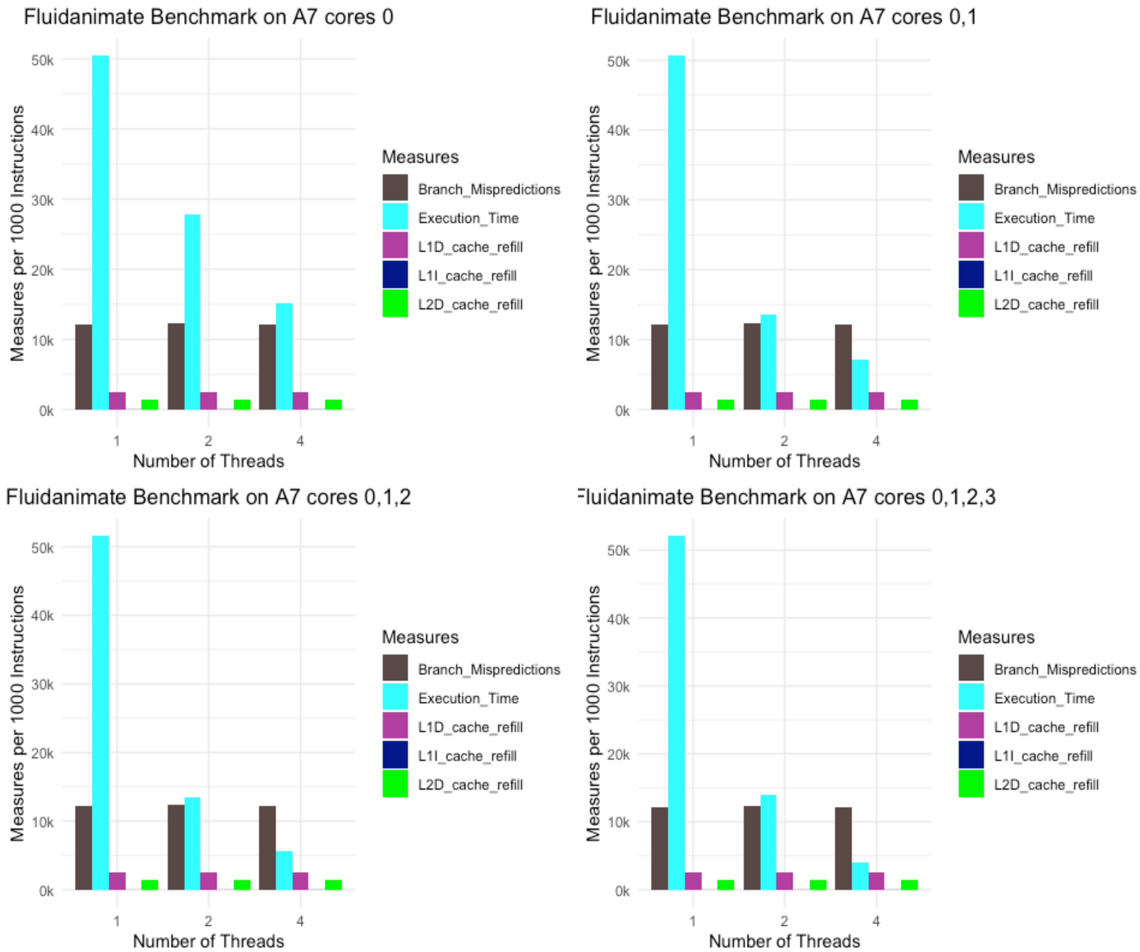

Figure 6: Fluidanimate A7

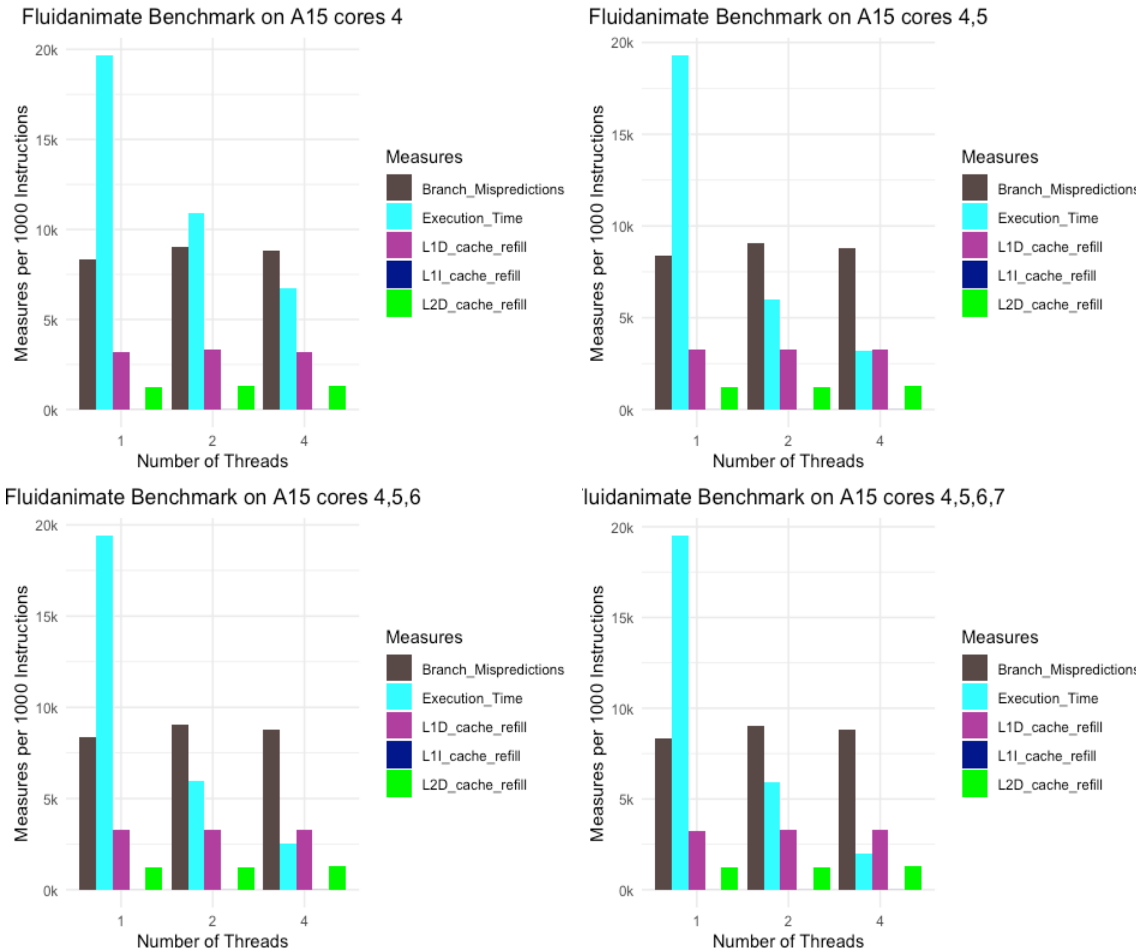

Figure 7: Fluidanimate A15

2. <u>BLACKSCHOLES:</u> With the blackscholes benchmark test, the observations are shown with the fluidanimate benchmark.

    a. The throughput was also better when we tested the program against four cores and four threads. It seems these patterns occurred with both core types; ARM7 (0,1,2,3) and ARM15(4,5,6,7). This suggests that the higher the cores-threads, the faster the execution time. For example, ARM 7: 4 threads and 4,5,6,7 (4 cores) used about 0.5 seconds to execute the exact instructions "4 threads and 4 (1 core)" executed in 1.2 seconds. The trend observed from the data points suggests that the more cores and threads, the faster the execution time. This means the system could perform parallel instruction execution (because of the threads) on multiple cores (increased core number).

    b. The branch misprediction rate was higher in the ARM 7 (13) core type than on the ARM 15(3) core type. Better branch prediction means less time wasted speculatively executing instructions that never actually need to be executed. This was why the ARM 15 cores (4 threads 4 cores) performed better (0.19secs) in terms of execution times than ARM 7 (4 threads 4 cores) (0.54secs).

    c. L1 and L2 cache refill rates were also almost constant across the two core types.

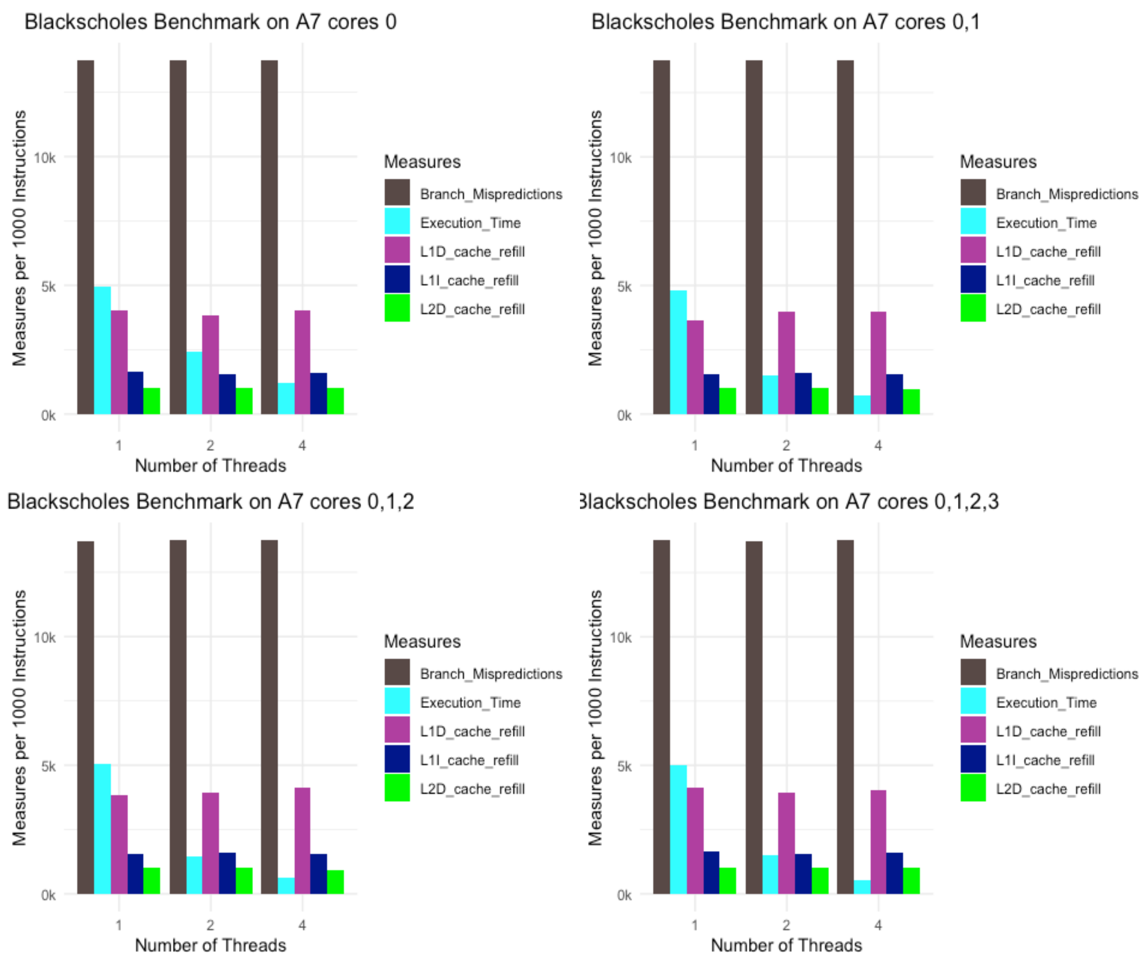

Figure 8: Blackscholes A7

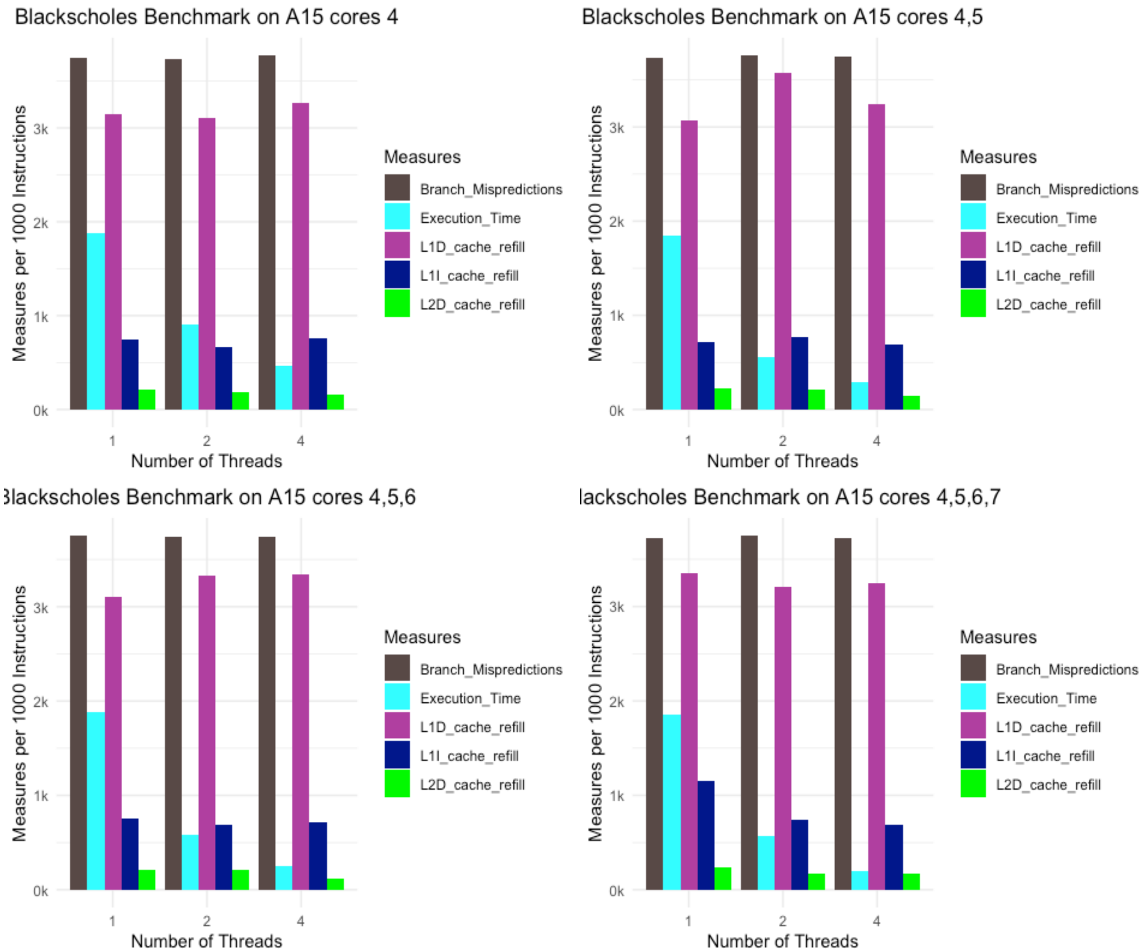

Figure 8: Blackscholes A15

3. BASICMATHS: With this benchmark test, there were a couple of observations:

    a. Varying (increasing) the cores did not increase or reduce the execution times. However, the execution time on ARM7(2.0) per core was more than the execution time on ARM15(0.9) per core. It seems the CPI on ARM 15 was faster.

    b. The branch misprediction rate did not vary much with varying core numbers. However, the ARM7 (22.4) misprediction rate was way higher than the ARM15 (9).

    c. The L1 and L2 cache refills were almost the same across the cores. Also, the ARM 15 had more L1 Data Cache refill when tested with four cores than the others.

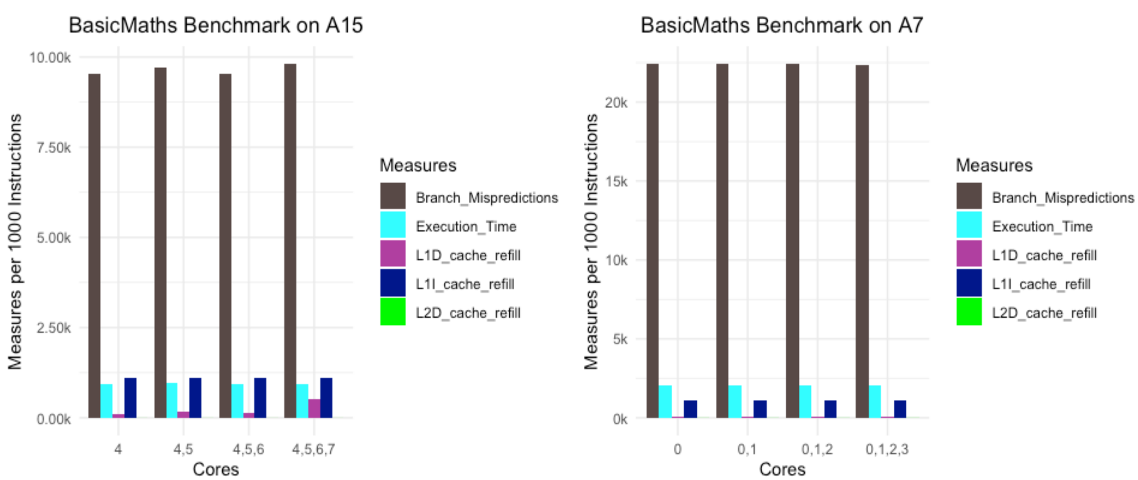

Figure 9: Basicmaths A7 & A15

4. QSORT: With the QSORT benchmark test, there were a couple of observations:

    a. Varying (increasing) the cores did not increase or reduce the execution times. However, the execution time on ARM7 per core was more than the execution time on ARM15 per core.

    b. The branch misprediction rate did not vary much with varying core numbers. However, the ARM7 misprediction rate was way higher than the ARM15.

    c. The L1 and L2 cache refills had almost the same value across the core numbers. However, there were more L1 data cache refills on the ARM15 than on the ARM7.

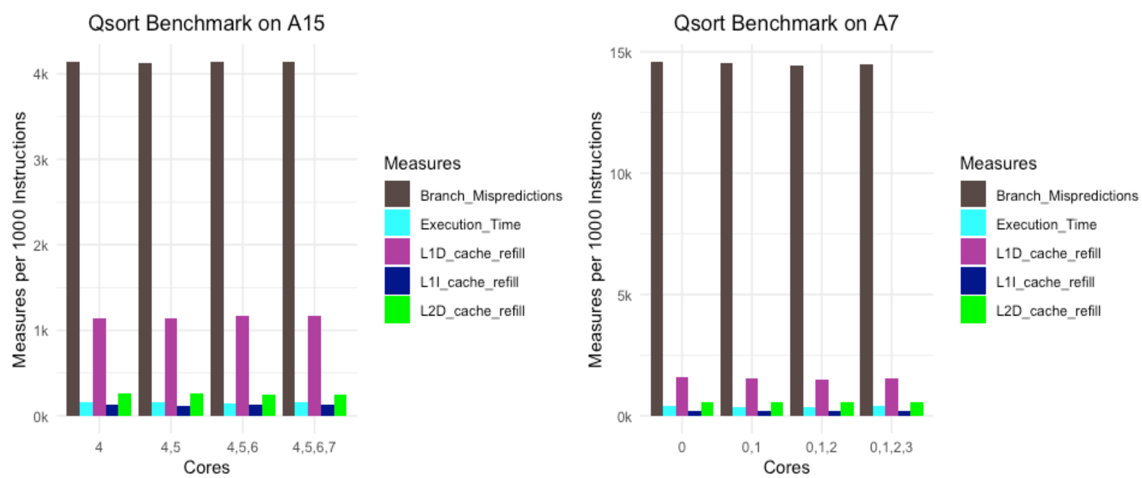

Figure 10: Qsort A7 & A15

APPENDIX

DATA POINTS

Core Type	Thread Number	Core Number	Instruction	Cycles	Branch_Mispredictions	L1D_cache_refill	L1I_Cache_Refill	L2D_Cache_Refill	Execution_Time	Benchmark
A15	1	4	36901083517	37247670618	8.354590957	3.199179864	0.003932649	1.221378942	19.66440152	Fluidanimate
	1	4,5	36901083825	36486851265	8.355099518	3.23959697	0.004350496	1.220741028	19.29514671	Fluidanimate
	1	4,5,6	36901084069	36951749439	8.349326741	3.309231967	0.003740414	1.226405379	19.41218882	Fluidanimate
	1	4,5,6,7	36901084403	37116575223	8.349693882	3.228286637	0.004120412	1.218014187	19.50911586	Fluidanimate
	2	4	37850985190	41606412229	9.037532602	3.313639976	0.007466825	1.30073038	10.92237052	Fluidanimate
	2	4,5	37817488252	40993473089	9.031605635	3.282583237	0.007359532	1.247932271	5.970440329	Fluidanimate
	2	4,5,6	37820293994	41253602556	9.024333295	3.315097146	0.006971918	1.253487374	5.977540185	Fluidanimate
	2	4,5,6,7	37822589743	40577087170	9.011867387	3.287523783	0.006542959	1.251151714	5.90140754	Fluidanimate
	4	4	38941094053	51715235877	8.787619566	3.182974217	0.010211064	1.344214647	6.7663293	Fluidanimate
	4	4,5	38905653522	45034712189	8.81165565	3.270780709	0.008945315	1.304665091	3.196727626	Fluidanimate
	4	4,5,6	38961767408	45730874083	8.802312219	3.251687225	0.009013426	1.319661275	2.515470583	Fluidanimate
	4	4,5,6,7	38764045774	46386267807	8.845893383	3.269424595	0.008195008	1.318958268	1.964239827	Fluidanimate
A7	1	0	36901402161	22165168086	12.19874231	2.469356185	0.025684453	1.415269238	50.4904919	Fluidanimate
	1	0,1	36899025775	23785702131	12.20540732	2.472982726	0.01551985	1.41495098	50.68475133	Fluidanimate
	1	0,1,2	36900460888	23627668365	12.20747467	2.485004806	0.015626273	1.416985084	51.61768111	Fluidanimate
	1	0,1,2,3	36900932965	24295724282	12.20295293	2.508748386	0.015862724	1.4191073	52.12928668	Fluidanimate
	2	0	37852533378	26374823935	12.28506827	2.52629401	0.033120584	1.44821191	27.7807482	Fluidanimate
	2	0,1	37805566820	23432925630	12.2953183	2.507236913	0.022326932	1.426433174	13.65216853	Fluidanimate
	2	0,1,2	37782136901	23331236372	12.29739724	2.531155457	0.022671507	1.430951082	13.51675696	Fluidanimate
	2	0,1,2,3	37795803135	23380898441	12.29337911	2.534864625	0.023805192	1.429324048	13.97986552	Fluidanimate
	4	0	38942834592	27912617585	12.15867793	2.485079793	0.032748164	1.468026367	15.11345493	Fluidanimate
	4	0,1	38890053595	25664997341	12.1635524	2.510662367	0.027338027	1.471247745	7.201298944	Fluidanimate
	4	0,1,2	38945570053	25146065730	12.14575728	2.487962299	0.026640617	1.47833269	5.545916585	Fluidanimate
	4	0,1,2,3	38714686890	24631052154	12.21455825	2.510082206	0.02634703	1.50150885	4.022088818	Fluidanimate
A15	1	4	2895980742	3632525595	3.749853895	3.139922699	0.742569625	0.212172682	1.872647518	Blacksholes
	1	4,5	2895981048	3598659059	3.738826378	3.07405131	0.712205858	0.222445286	1.847720442	Blacksholes
	1	4,5,6	2895981351	3631602667	3.753234689	3.10065947	0.758511445	0.207913286	1.877361801	Blacksholes
	1	4,5,6,7	2895981760	3620207306	3.727264383	3.347955018	1.147638559	0.237457412	1.857045703	Blacksholes
	2	4	2895982954	3555200087	3.738742655	3.109709487	0.662507928	0.186862058	0.91166834	Blacksholes
	2	4,5	2895983260	3570454671	3.760324452	3.576012153	0.770198052	0.207478409	0.562969807	Blacksholes
	2	4,5,6	2895983563	3618498523	3.739904974	3.334444224	0.691725381	0.204579775	0.5795053	Blacksholes
	2	4,5,6,7	2895983955	3559125874	3.751042536	3.212109187	0.743606215	0.170310796	0.562796888	Blacksholes
	4	4	2895987426	3603823130	3.772351554	3.263084609	0.752982436	0.157533948	0.466303628	Blacksholes
	4	4,5	2895987732	3587410511	3.749910452	3.244721042	0.689557709	0.143983115	0.285197499	Blacksholes
	4	4,5,6	2895988076	3590101843	3.740017701	3.339032164	0.719113458	0.123773069	0.245282027	Blacksholes
	4	4,5,6,7	2895988384	3542531618	3.718792655	3.25069628	0.682838144	0.168605073	0.192227807	Blacksholes
A7	1	0	2895703130	2327188298	13.73198468	4.048556479	1.625095572	0.994277568	4.934777795	Blacksholes
	1	0,1	2894830686	2283258010	13.75041813	3.650313316	1.571206227	0.997567612	4.823753895	Blacksholes
	1	0,1,2	2898325736	2346178270	13.71998064	3.812334432	1.555429379	0.995786279	5.031252904	Blacksholes
	1	0,1,2,3	2899317134	2365177310	13.76470556	4.134026087	1.645493098	1.007751779	5.029360524	Blacksholes
	2	0	2898541861	2327190479	13.73458331	3.843077152	1.547035676	0.992514445	2.431166291	Blacksholes
	2	0,1	2895800033	2303286377	13.7567411	4.003118609	1.580690177	0.996099627	1.479614891	Blacksholes
	2	0,1,2	2897878119	2244262723	13.74420203	3.91193149	1.589930682	0.995808616	1.451082122	Blacksholes
	2	0,1,2,3	2900609880	2291205606	13.72373213	3.930384461	1.536379538	0.995587521	1.519768988	Blacksholes
	4	0	2899456597	2221572088	13.72840508	4.035587455	1.584722233	0.991000292	1.22362411	Blacksholes
	4	0,1	2897492693	2201853389	13.74893683	3.996755204	1.573726603	0.944128352	0.720553835	Blacksholes
	4	0,1,2	2896096100	2300505049	13.75284738	4.136992093	1.542370319	0.93377714	0.616720996	Blacksholes
	4	0,1,2,3	2896159375	2497368740	13.75915543	4.041492365	1.587666195	0.999054826	0.537259954	Blacksholes
A15		4	1884218125	1695861359	9.529343814	0.105135563	1.11107943	0.002333417	0.94782332	BasicMaths
		4,5	1884218429	1742835324	9.697883776	0.15198521	1.112416144	0.002465213	0.98028653	BasicMaths
		4,5,6	1884218734	1681068082	9.549220238	0.126446749	1.094468473	0.002297681	0.933669626	BasicMaths
		4,5,6,7	1884219081	1705507060	9.809742147	0.519376264	1.096803102	0.002231517	0.942828724	BasicMaths
A7		0	1886857026	971116034.3	22.38507639	0.079050328	1.07168604	0.010445943	2.063047139	BasicMaths
		0,1	1885421919	997885483.7	22.40147466	0.070521262	1.069842766	0.010181099	2.085706989	BasicMaths
		0,1,2	1884928680	947490835.7	22.42015314	0.074009343	1.06769946	0.009782156	2.083836922	BasicMaths
		0,1,2,3	1886461463	951189417.7	22.37513929	0.100804074	1.067988952	0.010431877	2.037007755	BasicMaths
A15		4	302512314	276868821.7	4.139711373	1.145047603	0.122540466	0.259174023	0.158164156	Qsort
		4,5	302512618	273817090	4.132812734	1.143346248	0.122052209	0.257107733	0.155222211	Qsort
		4,5,6	302512923	268101413	4.133625062	1.16631932	0.122881803	0.252652567	0.151613169	Qsort
		4,5,6,7	302513270	273519743	4.134390534	1.172275848	0.123122974	0.252362483	0.154964934	Qsort
A7		0	303202170.7	166049473	14.58831904	1.56836718	0.223841845	0.544447509	0.40326281	Qsort
		0,1	301564584.3	173450669.7	14.51821565	1.562198253	0.203357213	0.547819633	0.372029741	Qsort
		0,1,2	302223931.3	155734308	14.43885327	1.506248467	0.203168336	0.554174072	0.339112963	Qsort
		0,1,2,3	304052370.3	188279207.7	14.49199358	1.531817253	0.203358608	0.535055852	0.409313131	Qsort