International Journal of Scientific Research in Computer Science and Engineering

Research Paper
Volume-1, Issue-1, Jan- Feb-2013
Available online at www.isroset.org

ISSN: 2320 - 7639

Comparison of Intel Single-Core and Intel Dual-Core Processor Performance

A. Fasiku, Ayodeji Ireti[1], B. Olawale, Jimoh Babatunde[2], C. Abiola Oluwatoyin B.[3]

[1] Computer Engineering Department, Ekiti State University, Ado - Ekiti, Nigeria. iretiayous76@yahoo.com
[2] Computer Engineering Department, Rufus Giwa Polytechnic, Owo, Nigeria, jimohol@yahoo.co.uk
[3] Computer Science Department, Afe Babalola University, Ado - Ekiti, Nigeria. adeoyetoyin@yahoo.com

***Abstract* - Computer System Performance and Evaluation deal with investigation of computer components (both hardware and software) with a view of establishing the level of their performances. This research work, carry out performance evaluation studies on Intel single-core and dual-core process to know which of the processor have better execution time and throughput. The architecture of Intel single-core and Intel duo-core processor were studied. Dual-core processors deliver better performance-to-cost ratios relative to their single-core predecessors through on-chip multi-threading. However, they present challenges in developing high performance multi-threaded code. SPEC CPU2006 benchmarks suite was used to measure the performance of the processors. The overall execution time and throughput measurement of Intel single and dual core processors was reported and compared to each other. The research results showed that the execution time of CQ56 Intel Pentium Dual-Core Processor is about 3.42% faster than Intel Celeron Single Core Processor while the throughput of Intel Pentium Dual-Core Processor was found to be 1.03 times higher than Intel Celeron Single Core Processor.**

***Keywords*- Intel Dual-Core Processor, Intel Single-Core Processor, architecture and SPEC CPU2006**

1 INTRODUCTION

Modern microprocessors are among the most complex systems ever created by humans. A single silicon chip, roughly the size of a fingernail, can contain a complete high-performance processor, large cache memories, and the logic required to interface it to external devices. In terms of performance, the processors implemented on a single chip today dwarf the room-sized supercomputers that cost over $10 million just 40 years ago. Even the embedded processors found in everyday appliances such as cell phones, personal digital assistants, and handheld game systems are far more powerful than the early developed computers ever envisioned.

There has been an ever increasing demand for higher and higher processing speeds. However, in this growing competition of making processors faster and faster, CPU designers have nearly exhausted their collective bag of tricks to get more performance out of additional transistors on a chip by increasing parallelism at the instruction level. Speculative execution and deep pipelining are by now very standard features and CPU designs are getting increasingly complex and hard to manage. This problem is further exacerbated by the fact that chips are sucking up large amounts of power and expending much of it as heat and the problem grows more acute as clock speeds ramp up. However, AMD, Intel and other processors manufacturers have found solution to this problem by redesigning the processor architecture differently and introduce multi-core processors [4]. The performance of computer systems have been doubling every year for more than 40years, a computer today with high performance processor is a thousand times more powerful than the one built in the early 70's to mid 70's. Technological improvements have been fairly steady, progress arising from better computer architectures has been consistent and sustaining the recent improvements in cost and performance will require continuous innovations in computer design [3], [7].

It is well-recognized fact that computer processors have increased in speed and decreased in cost at a tremendous rate for a very long time. This observation was first made popular by Gordon Moore in 1965, and is commonly referred to as the Moore's Law. Specifically, Moore's Law states that the advances in electronic manufacturing technology make it possible to double the number of transistors per unit area about every 12 to 18 months. It is this advancement that has fueled the phenomenal growth in computer speed and accessibility over more than five decades. Smaller transistors have made it possible to increase the number of transistors that can be applied to processor functions and reduce the distance signals must travel, allowing processor clock frequencies to soar. This simultaneously increases system performance and reduces system cost [6].

Over the years, there has been a growing interest in designing microprocessor based on the notion of Instruction Level Parallelism (ILP). There are different approaches for exploiting ILP. One approach uses run – time scheduling to evaluate the data dependences and execute instructions concurrently. A microprocessor based on this technique is called a superscalar microprocessor. Another approach, commonly known as Very Long Instruction Word (VLIW) architecture, is entirely based on compile – time analysis to extract parallelism. Superscalar architectures and VLIW architectures both improve processor performance by increasing the CPI (Cycle Per Instruction) factor. These architectures exploit ILP by issuing more than one instruction per cycle. VLIW processors require a sophisticated compiler to extract ILP prior to execution, which results in code expansion [1]. This leads to the development of Thread Level Parallelism (TLP), which is explicitly represented by the use of multiple threads of execution that are inherently parallel.

Increasing the performance of single-threaded processors becomes increasingly difficult and one way to enhance the performance of chip multiprocessors that has received considerable attention is the use of thread-level parallelism [9]. Therefore, a thread is a flow of control through a program with a single execution point. In order word, a thread is a separate process with its own instructions and data. A thread may represent a process that is part of a parallel program consisting of multiple processes, or it may represent an independent program on its own. Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute. Unlike instruction-level parallelism, which exploits implicit parallel operations within a loop or straight-line code segment, thread-level parallelism is explicitly represented by the use of multiple threads of execution that are inherently parallel.

2 Architecture of Single Core Processor

The last 30 years have seen the computer industry driven primarily by faster and faster uniprocessors; those days have come to a close. Emerging in their place are microprocessors containing multiple processor cores that are expected to exploit thread-level parallelism. Pentium 4 processor brand, refers to Intel's line of single-core desktop and laptop Central Processing Units (CPUs) introduced on November 20, 2000 at a speed rate of 1.5GHZ and shipped through August 8, 2008. They are the 7th-generation X86 micro architecture, called NetBurst, which was the company's latest design of single core processor designed since the introduction of P6 micro architecture of the Pentium Pro CPUs in 1995. NetBurst differed from the preceding P6 (Pentium III, II, etc.), featuring a very deep instruction pipeline to achieve very high clock speeds (up to 3.8 GHz). The first Pentium 4 cores, codenamed Willamette, were clocked from 1.3 GHz to 2 GHz, Pentium 4 CPUs introduced the SSE2 and, in the Prescott-based Pentium 4's, Streaming SIMD Extensions 3 (SSE3) instruction sets to accelerate calculations, transactions, media processing, 3D graphics, and games [8].
This processor requires balancing and tuning of many microarchitectural features that compete for processor die cost and for design and validation efforts. Figure 1 shows the basic Intel NetBurst microarchitecture of the Pentium 4 processor. As you can see, there are four main sections in this architectural diagram of Pentium 4 as show in figure 1.

a) The in-order front end
b) The out-of-order execution engine
c) The integer and floating-point execution units
d) The memory subsystem

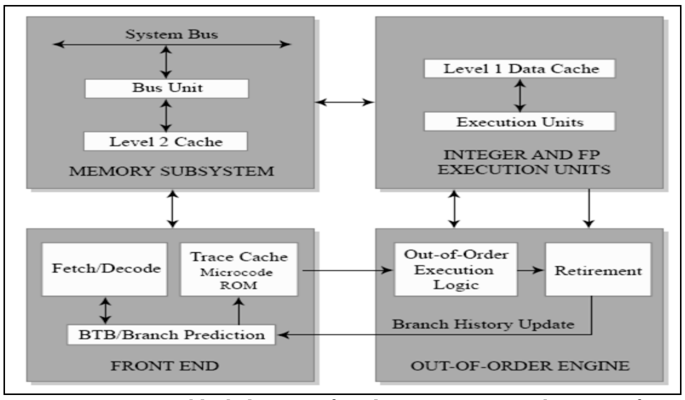


Figure 1: Basic block diagram of Intel NetBurst microarchitecture of Pentium 4 [2]

a) In-Order Front End

The in-order front end is part of the machine that fetches the instructions to be executed next in the program and prepares them to be used later in the machine pipeline. Its job is to supply a high-bandwidth stream of decoded instructions to the out-of-order execution core, which will do the actual completion of the instructions. The front end has highly accurate branch prediction logic that uses the past history of program execution to speculate where the program is going to execute next. The predicted instruction address, from this front-end branch prediction logic, is used to fetch instruction bytes from the Level 2 (L2) cache. These IA-32 instruction bytes are then decoded into basic operations called micro-operations (micro-operations) that the execution core is able to execute [11].

The NetBurst micro architecture has an advanced form of a Level 1 (L1) instruction cache called the Execution Trace Cache. Unlike conventional instruction caches, the Trace Cache sits between the instructions decode logic and the execution core as shown in Figure 3.1. In this location the Trace Cache is able to store the already decoded IA-32 instructions or micro-operations (uops). Storing already decoded instructions removes the IA-32 decoding from the main execution loop. Typically the instructions are decoded once and placed in the Trace Cache and then used repeatedly from there like a normal instruction cache on previous machines. The IA-32 instruction decoder is used only when the machine misses the Trace Cache and needs to go to the L2 cache to get and decode new IA-32 instruction bytes.
The front-end decoder translates each IA-32 instruction to a series of micro-operations (uops), which are similar to typical RISC instructions. The micro-operations are than executed by a dynamically scheduled speculative pipeline. A trace cache is a type of instruction cache that holds sequences of instructions to be executed including nonadjacent instructions separated by branches; a trace cache tries to exploit the temporal sequencing of instruction execution rather than the spatial locality exploited in a normal cache.

The Pentium 4's execution trace cache is a trace cache of micro-operations, corresponding to the decoded IA-32 instruction stream. By filling the pipeline from the execution trace cache, the Pentium 4 avoids the need to red code IA-32 instructions whenever the trace cache hits. Only on trace cache misses are IA-32 instructions fetched from the L2 cache and decoded to refill the execution trace cache. Up to three IA-32 instructions may be decoded and translated every cycle, generating up to six micro-operations; when a single IA-32 instruction requires more than three micro-operations, the micro-operations sequence is generated from the microcode ROM. The execution trace cache has its own branch target buffer, which predicts the outcome of micro-operations branches. After fetching from the execution trace cache, the micro-operations are executed by an out-of-order execution [8].

b) Out-Of-Order Execution Logic

The out-of-order execution engine is where the instructions are prepared for execution. The out-of-order execution logic has several buffers that it uses to smooth and re-order the flow of instructions to optimize performance as they go down the pipeline and get scheduled for execution. Instructions are aggressively reordered to allow them to execute as quickly as their input operands are ready. This out-of-order execution allows instructions in the program following delayed instructions to proceed around them as long as they do not depend on those delayed instructions.

Out-of-order execution allows the execution resources such as the ALUs and the cache to be kept as busy as possible executing independent instructions that are ready to execute.
The retirement logic is what reorders the instructions, executed in an out-of-order manner, back to the original program order. This retirement logic receives the completion status of the executed instructions from the execution units and processes the results so that the proper architectural state is committed (or retired) according to the program order. The Pentium 4 processor can retire up to three micro-operations per clock cycle. This retirement logic ensures that exceptions occur only if the operation causing the exception is the oldest, non-retired operation in the machine. This logic also reports branch history information to the branch predictors at the front end of the machine so they can train with the latest known-good branch-history information [11].

c) Memory Subsystem

The memory subsystem showed in figure 3.1, includes the L2 cache and the system bus. The L2 cache stores both instructions and data that cannot fit in the Execution Trace Cache and the L1 data cache. The external system bus is connected to the backside of the second-level cache and is used to access main memory when the L2 cache has a cache miss, and to access the system I/O resources.

d) Integer and Floating-Point Execution Units

The execution units are where the instructions are actually executed. This section includes the register files that store the integer and floating-point data operand values that the instructions need to execute. The execution units include several types of integer and floating-point execution units that compute the results and also the L1 data cache that is used for most load and store operations.

3 MULTICORE ARCHITECTURES

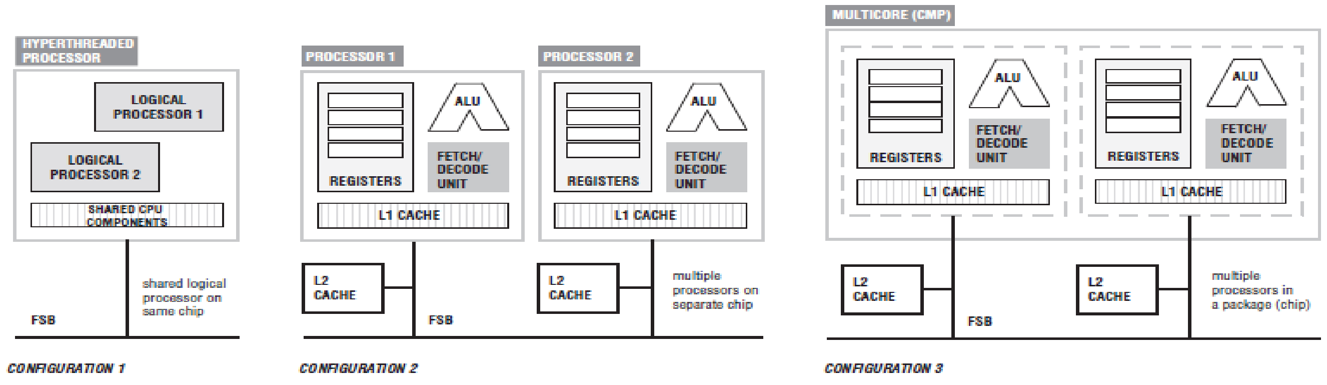


Figure 2: Architectural Configuration of Multiprocessing

Chip Multiprocessors (CMPs) come in multiple flavours: two processors (dual core), four processors (quad core), and eight processors (octa - core) configurations. Some configurations are multithreaded while some are not. There are several variations in how cache and memory are approached in the new Chip Multiprocessors. Again, most of these differences are not visible when looking strictly at the logical view of an application that is being designed to take advantage of a multicore architecture. Figure 2, illustrates three common architectural configurations that support multiprocessing [5].

a) Configuration 1 in Figure 2, uses hyperthreading. Like Chip Multiprocessors, an hyperthreaded processors allows two or more threads to execute on a single chip. However, in a hyperthreaded package the multiple processors are logical instead of physical. There are some duplication of hardware but not enough to qualify a separate physical processor. So hyperthreading allows the processor to present itself to the operating system as complete multiple processors when in fact there is a single processor running multiple threads.
b) Configuration 2 in Figure 2, is the classic multiprocessor. In configuration 2, each processor is on a separate chip with its own hardware.
c) Configuration 3 represents the current trend in multiprocessors. It provides complete processors on a single chip.

What important to remember is that each configuration presents itself to the developer as a set of two or more logical processors capable of executing multiple tasks concurrently. The challenge for system programmers, kernel programmers, and application developers is to know when and how to take advantage of it.

4 BENCHMARKS SUITES

Benchmarks and metrics to be used for performance evaluation have always been interesting for system developers because they have access to different types. There has been a lot of improvement in benchmark suites since 1988. SPEC originally created a benchmark set focusing on processor performance (initially called SPEC89), which has evolved into its fifth generation: SPEC CPU2006, which follows SPEC2000, SPEC95, SPEC92, and SPEC89. SPEC CPU2006 consists of a set of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks
(CFP2006) [10]. Before, computer performance evaluation have been with small benchmarks such as kernels extracted from applications such as Lawrence Livermore Loops, Dhrystone and Whetstone benchmarks, Linpack, Sorting, Sieve of Eratosthenes, 8-queens problem, Tower of Hanoi, etc. [5].The Standard Performance Evaluation Corporation (SPEC) consortium and Transactions Processing Council (TPC) formed in 1980s have made available several benchmark suites and benchmarking guidelines to improve the quality of benchmarking.

The best choices of benchmarks to measure performance are real applications, such as a compiler. Attempts at running programs that are much simpler than a real application have led to performance pitfalls. Examples include:

a) Kernels, which are small, key pieces of real applications;
b) Toy programs, Toy benchmark are typically between 10 and 100 lines of code and produce a result the user already knows before running the toy program.
c) Synthetic benchmarks, which are fake programs invented to try to match the profile and behaviour of real applications, such as Dhrystone.

All these three are discredited today, usually because the compiler writer and architect can conspire to make the computer appear faster on these stand-in programs than on real applications [8].

The key issues that benchmark designers face in deciding modification of the source is whether such modifications will reflect real practice and provide useful insight to users or whether such modifications simply reduce the accuracy of the benchmarks as predictors of real performance. To overcome the danger of placing too many eggs in one basket, collections of benchmark applications, called benchmark suites, are a popular measure of performance of processors with a variety of applications. Of course, such suites are only as good as the constituent individual benchmarks. Nonetheless, a key advantage of such suites is that the weakness of any one benchmark is lessened by the presence of the other benchmarks [10].

5 MEASURING PERFORMANCE

When we say one computer is faster than another, what we are actually saying is that a program runs in less time on one computer than the other i.e the user of a desktop or laptop computer may say a computer is faster when a program runs in less time. Computer user is actually interested in reducing response time or execution time (the time between the start and the completion of an event) but interested in increasing the throughput (the total amount of work done in a given time).

In comparing design alternatives, it is often the practice to relate the performance of two different computers, say, X and Y. When computer X is faster than computer Y, this means that the response time or execution time is lower on X than on Y for the given task. Then, computer X is n times faster than computer Y.

Therefore, $\frac{Execution\ time\ of\ Y}{Execution\ time\ of\ X} = n$ ------------- (1)

Since execution time is the reciprocal of performance, therefore the following relationship holds:

$$n = \frac{Execution\ time\ of\ Y}{Execution\ time\ of\ X} = \frac{\frac{1}{Performance\ of\ Y}}{\frac{1}{Performance\ of\ X}} = \frac{Performance\ of\ X}{Performance\ of\ Y} \text{ ----- (2)}$$

The phrase "the throughput of X is 1.3 times higher than Y" signifies here that the number of tasks completed per unit time on computer X is 1.3 times the number completed on Y. The only consistent and reliable measure of performance is the execution time of real programs [8]. Performance and execution time are reciprocals, increasing performance decreases execution time.

Benchmark programs used in this research for measurement of processors performance is SPEC CPU2006, SPEC designed CPU2006 to provide a comparative measure of compute-intensive performance across the widest practical range of hardware using workloads developed from real user applications. These benchmarks are provided as source code and require the user to be comfortable using compiler commands as well as other commands via a command interpreter using a console or command prompt window in order to generate executable binaries. The current version of the benchmark suite is V1.2, released in September, 2011. The SPEC CPU benchmark suite is primarily used by computer system manufacturers to measure and report performance of their computer systems and by microprocessor designers and researchers for evaluating designs trade-offs and novel design ideas. Customers typically use the results reported to SPEC to make purchasing decisions.

SPEC CPU2006 developed twelve integers and seventeen floating-point-intensive application programs used to analyse/measure the performance of computer system. The SPEC CPU2006 benchmark consists of two benchmark suites, which focuses on a different aspect of compute-intensive performance. CINT2006 measure and compare compute-intensive integer performance, while CFP2006 measures and compares compute-intensive floating-point performance. Though, for the purpose of this research we limit our performance measurement to CINT2006 suite i.e we ran only the CINT2006 SPECint_base benchmark, which comprised of 12 integer-compute-intensive codes; 9 in C and 3 in C++. These programs (benchmarks in the CINT2006 suite) were run on Hewlet-Packard (HP) Intel Pentium(R) Dual-Core Processor and Toshiba Satellite C660, Intel Celeron Single-Core Processor to compare the execution time (speed) and throughput of the processors.

Table 1 summarizes some of the key aspects of the configurations of the systems tested i.e Toshiba Satellite C660, Intel Celeron Single-Core Processor and HP Presario CQ56 Intel Pentium Dual-Core Processor.

Table 1 Configurations of the two Systems Under Test (SUT)

SYSTEM	TOSHIBA SATELLITE C660, INTEL CELERON SINGLE-CORE PROCESSOR	HP PRESARIO CQ56 INTEL PENTIUM DUAL-CORE PROCESSOR
CPU		
Vendor	Intel	Intel
Model	Satellite C660 Notebook	Presario CQ56 Notebook
Name	Intel Celeron	Pentium (R) T4500
Core Frequency (GHZ)	2.30GHz	2.30GHz
Number of Processor Package	1	2
Number of Threads	1	2
Processor Type	Celeron Single Core Processor	Pentium (R) Dual-Core

Platform		
BIOS Name and Version	HP BIOS F.03	HP BIOS F.07
BIOS Setting	Default	Default
Memory Module(s)		
Vendor and Model Numbers	ECPIDA TAIWAN and EBJ21UE8BDSO-DJ-F	HYNIX KOREA 09 and HYMP125S64CP8-S6-AB
Type	PC3-10600S-9-10-F1	PC2-6400S-666-12
Size	4 GB	4 GB
Number of RAM Modules	2 X 2,048 MB	2 X 2,048 MB
Chip Organisation	Double-sided	Double-Sided
Hard Disk		
Vendor and Model Numbers	Seagate and ST93015A	Hitachi and HTB-TS5SAA500B
Type	Serial ATA-150 - 7200.0 rpm	SATA 3.0Gb/s TS5SAA320
Size	320 GB	320 GB
Number of Disk in System	1	1
Operating System		
Name	Fedoral 10.0	Fedoral 10.0
File System	EXT 3	EXT 3
Kernel	2.6.27.5-117.fc10.i686	2.6.27.5-117.fc10.i686
Language	English	English

6 DISCUSSIONS OF RESULTS

The actual test results consist of the execution times and ratios for the individual benchmarks and the overall SPEC metric produced by running the benchmarks via the SPEC tools. It required the use of the SPEC tools and ensures that the results generated are based on benchmarks built, run, and validated according to the SPEC run rules.

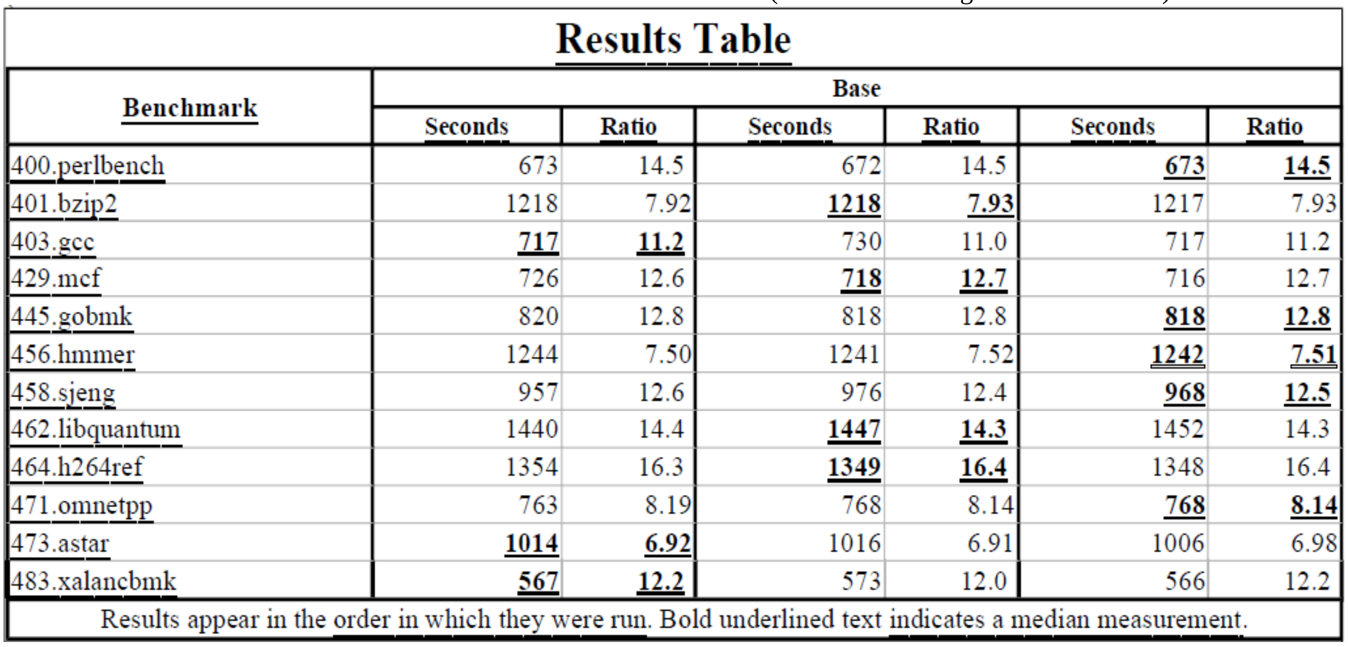
The execution times in seconds for each of the benchmarks in the CINT2006 suite are generated and the ratio of the System Under Test (SUT) are calculated. The SPECint_base2006 metrics are calculated as a Geometric Mean of the individual ratios, where each ratio is based on the median execution time from three runs.

The throughput metrics are calculated based on the execution of benchmark binaries that are built using the same rules as binaries built for speed metrics. However, the tester may select the number of concurrent copies of each benchmark to be run, the same number of copies must be used for all benchmarks. A CINT2006 run and performs each of the twelve applications (tasks) three times and also reports the median for each. It also calculated the geometric mean of those 12 applications to produce an overall score i.e the throughput.

a. Result of Execution Time of the Two Systems Under Test

The Execution time of Toshiba Satellite C660 (Intel Celeron Single Core Processor) generated from SPEC CINT2006 suite shown in table 2 and the graph in figure 3, while table 3 and figure 4 showed the Execution time and graph generated for HP Presario CQ56 Intel Pentium Dual-Core Processor. Execution time is the time between the start and the completion of an event i.e the process of carrying out an instruction within a specific time in any computer system. The median value from the three result generated is boldly underline in the table.

Table 2 The Execution times of Toshiba Satellite C660 (Intel Celeron Single Core Processor)

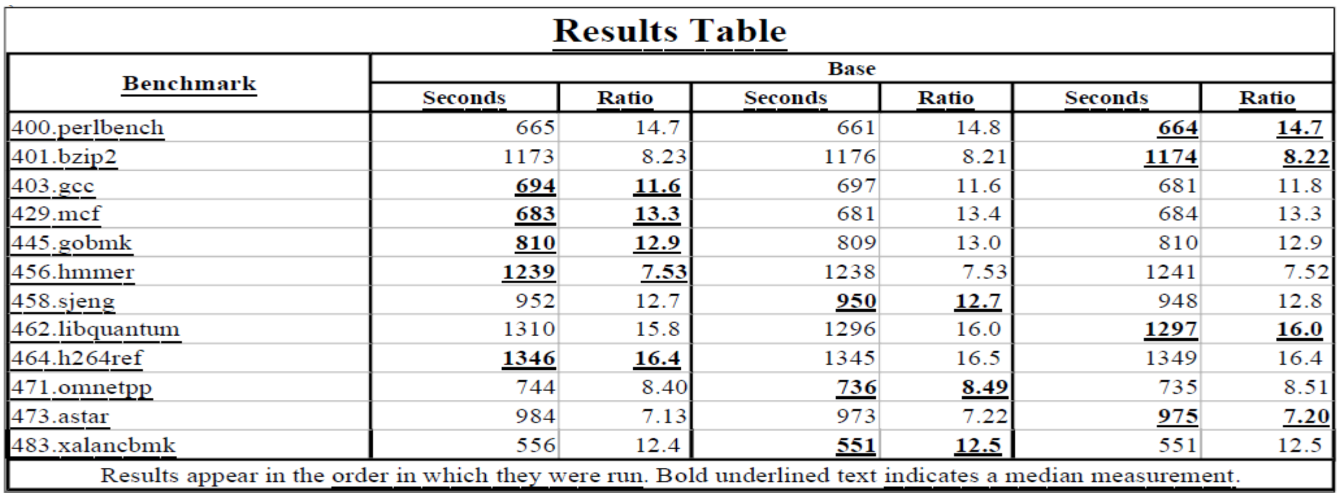


Results Table

Benchmark	Base					
	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
400.perlbench	673	14.5	672	14.5	**673**	**14.5**
401.bzip2	1218	7.92	**1218**	**7.93**	1217	7.93
403.gcc	**717**	**11.2**	730	11.0	717	11.2
429.mcf	726	12.6	**718**	**12.7**	716	12.7
445.gobmk	820	12.8	818	12.8	**818**	**12.8**
456.hmmer	1244	7.50	1241	7.52	**1242**	**7.51**
458.sjeng	957	12.6	976	12.4	**968**	**12.5**
462.libquantum	1440	14.4	**1447**	**14.3**	1452	14.3
464.h264ref	1354	16.3	**1349**	**16.4**	1348	16.4
471.omnetpp	763	8.19	768	8.14	**768**	**8.14**
473.astar	**1014**	**6.92**	1016	6.91	1006	6.98
483.xalancbmk	**567**	**12.2**	573	12.0	566	12.2

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.

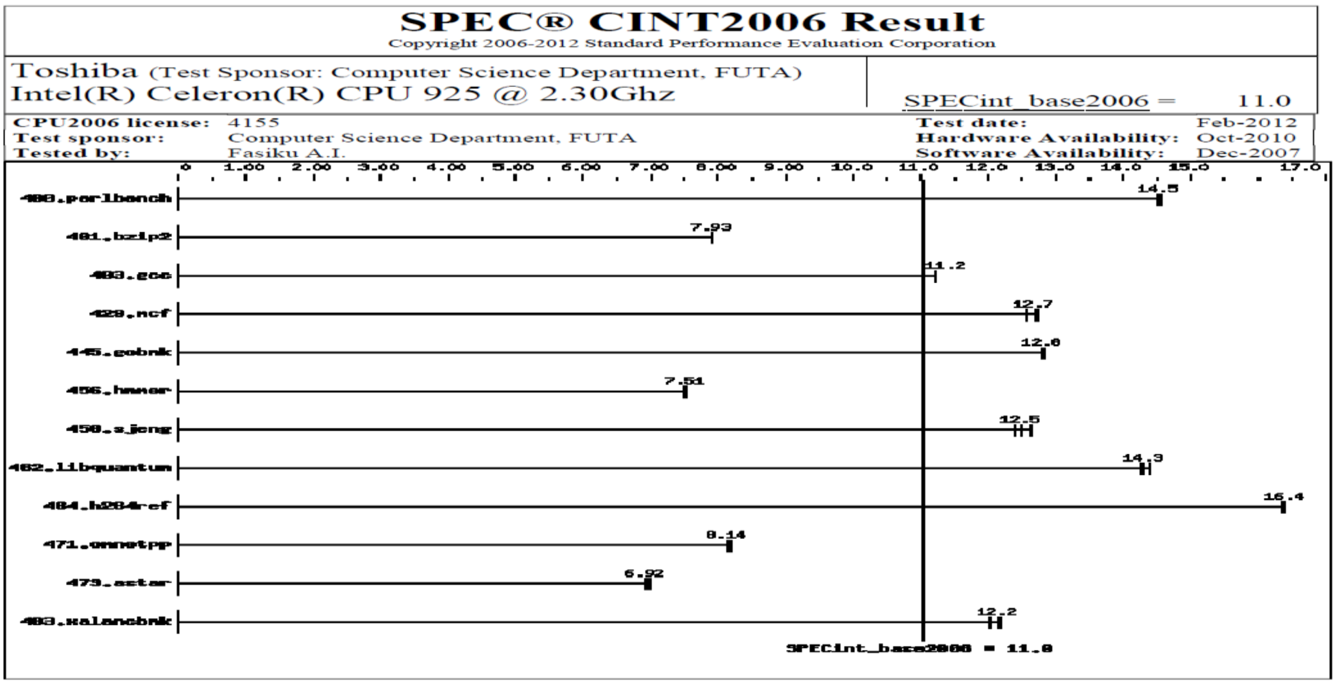
Table 3 The Execution times of HP Intel Pentium (R) Dual-Core Processor



Results Table

Benchmark	Base					
	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
400.perlbench	665	14.7	661	14.8	**664**	**14.7**
401.bzip2	1173	8.23	1176	8.21	**1174**	**8.22**
403.gcc	**694**	**11.6**	697	11.6	681	11.8
429.mcf	**683**	**13.3**	681	13.4	684	13.3
445.gobmk	**810**	**12.9**	809	13.0	810	12.9
456.hmmer	**1239**	**7.53**	1238	7.53	1241	7.52
458.sjeng	952	12.7	**950**	**12.7**	948	12.8
462.libquantum	1310	15.8	1296	16.0	**1297**	**16.0**
464.h264ref	**1346**	**16.4**	1345	16.5	1349	16.4
471.omnetpp	744	8.40	**736**	**8.49**	735	8.51
473.astar	984	7.13	973	7.22	**975**	**7.20**
483.xalancbmk	556	12.4	**551**	**12.5**	551	12.5

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.



SPEC® CINT2006 Result

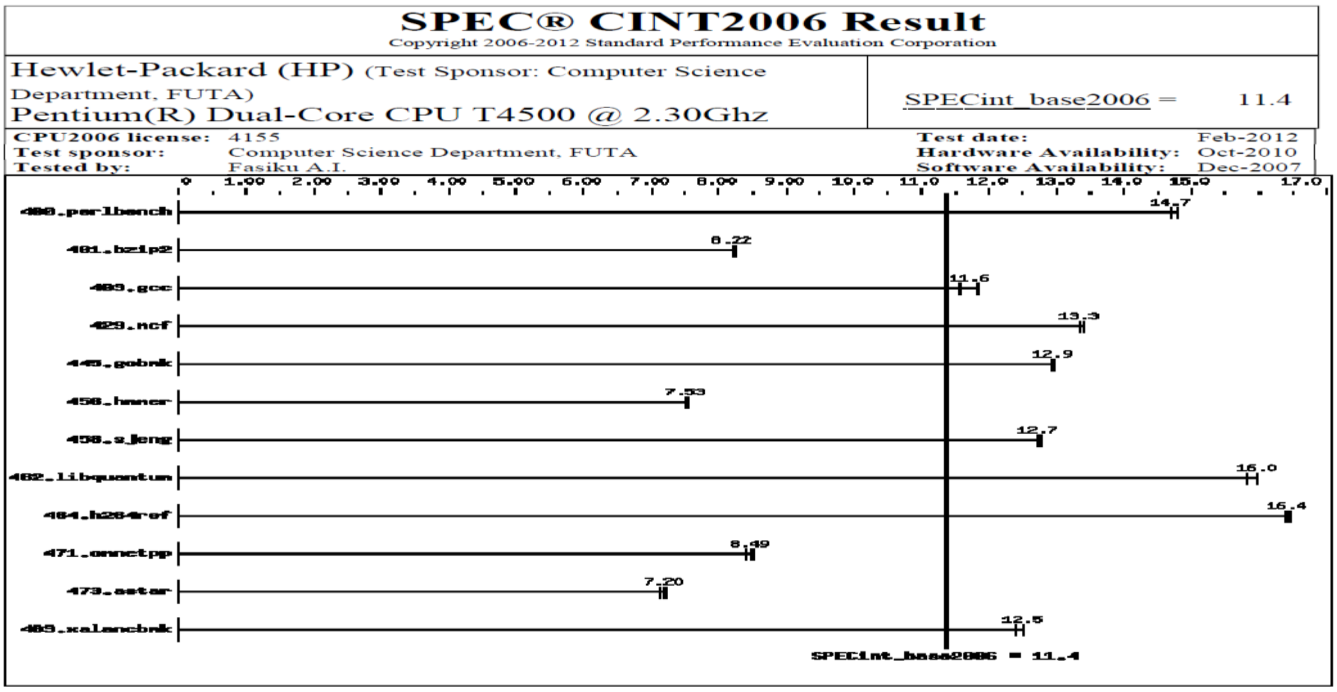
Copyright 2006-2012 Standard Performance Evaluation Corporation

Toshiba (Test Sponsor: Computer Science Department, FUTA)
Intel(R) Celeron(R) CPU 925 @ 2.30Ghz

SPECint_base2006 = 11.0

CPU2006 license:	4155	Test date:	Feb-2012
Test sponsor:	Computer Science Department, FUTA	Hardware Availability:	Oct-2010
Tested by:	Fasiku A.I.	Software Availability:	Dec-2007

Figure 3 Graph showing the result of Toshiba Satellite C660 (Intel Celeron Single Core Processor)



SPEC® CINT2006 Result

Copyright 2006-2012 Standard Performance Evaluation Corporation

Hewlet-Packard (HP) (Test Sponsor: Computer Science Department, FUTA)
Pentium(R) Dual-Core CPU T4500 @ 2.30Ghz

SPECint_base2006 = 11.4

CPU2006 license:	4155	Test date:	Feb-2012
Test sponsor:	Computer Science Department, FUTA	Hardware Availability:	Oct-2010
Tested by:	Fasiku A.I.	Software Availability:	Dec-2007

Figure 4: Graph showing the result of HP Intel Pentium (R) Dual-Core Processor

A. Comparing Intel Single and Dual Core Processor Execution Time

Table 2 and 3 showed the details results generated from running SPECint_base2006 on each system. The execution units depend on the number of benchmarks run in a given system under test. The median scores boldly underline in table 2 and 3, were picked from each system tested and compare the execution time as shown in the table 4. These values were used to compare the performance of the two processors tested and the charts layouts shown in figure 5, displays the difference in performance of the two processors tested.

Table 4: Intel Celeron Single-Core and Intel Dual-Core Processor Systems Execution times

	Comparing the Intel Celeron Single-Core and Intel Dual-Core Processors Execution times		
		SPEC CPU2006 Execution Time in seconds	
	Benchmarks	Intel Celeron Single Core	Intel Pentium Dual Core
1	400.perlbench	673	664
2	401.bzip2	1218	1174
3	403.gcc	717	694
4	429.mcf	718	683
5	445.gobmk	818	810
6	456.hmmer	1242	1239
7	458.sjeng	968	950
8	462.libquantum	1447	1297
9	464.h264ref	1349	1346
10	471.omnetpp	768	736
11	473.astar	1014	975
12	483.xalancbmk	567	551
	Total Execution Time	11499	11119

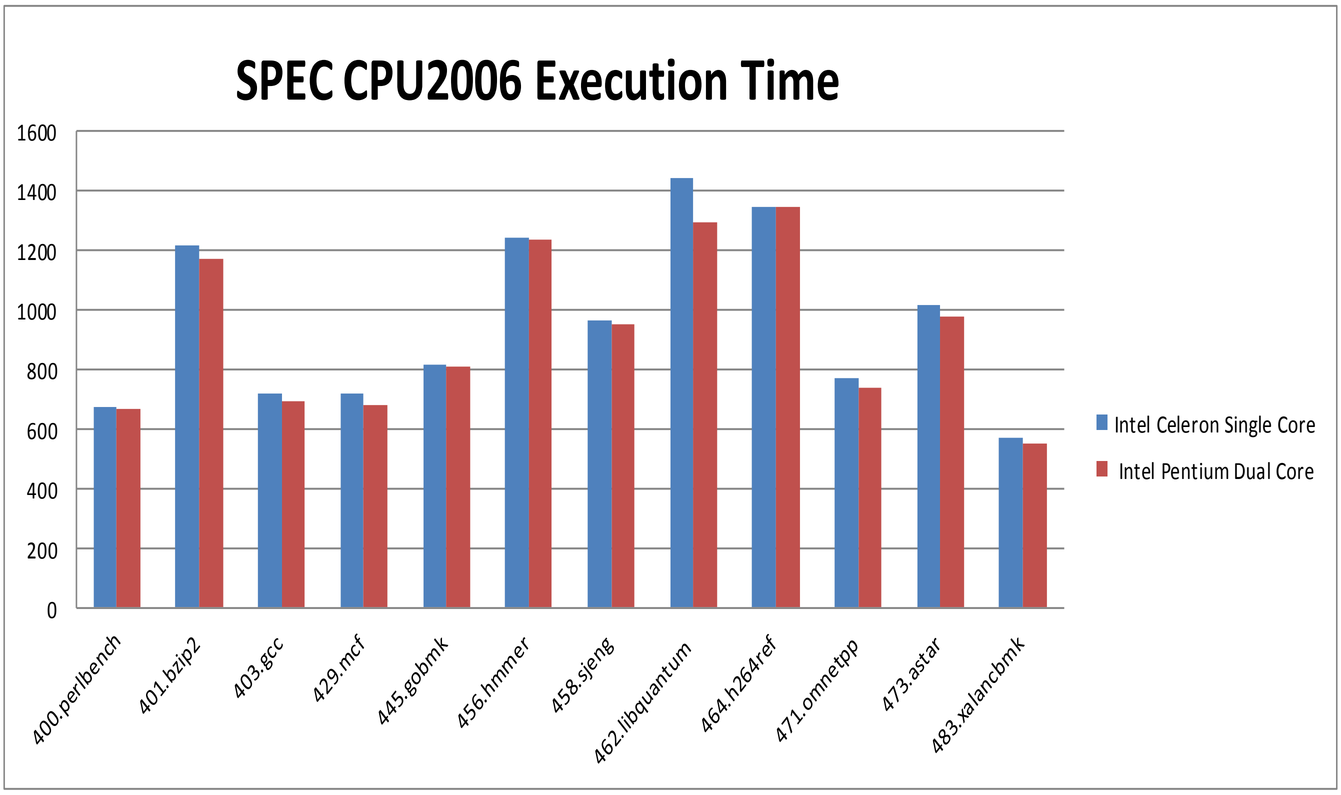


Figure 5: SPEC CPU Execution Time

The performance results of Intel Celeron Single Core Processor and Intel Pentium Dual-Core Processor generated from SPEC CINT2006 suite shown in table 4 and the total execution time for the processors were as follows:

Total Execution time of Intel Single Core Processor is 11499 seconds
Total Execution time of Intel Dual Core Processor is 11119 seconds
Given,

$$n = \frac{Execution\ time\ of\ Intel\ Single - Core\ Processor\ (Ex)}{Execution\ time\ of\ Intel\ Dual - Core\ Processor\ (Ey)}$$

------- (3)

To calculate the percentage of the processor execution time

$$\frac{Execution\ time\ of\ Intel\ Single{-}Core\ Processor\ (Ex)}{Execution\ time\ of\ Intel\ Dual{-}Core\ Processor\ (Ey)} = 1 + \frac{n}{100}$$

------- (4)

Therefore,

$$Ex(Intel\ Single\ Processor) =$$

$$Ex\ (Intel\ Dual\ Processor) + \frac{n{*}Ey(Dual\ Processor)}{100} \text{ --- (5)}$$

$$100\{\ Ex(Single\ processor) - Ex\ (Dual\ Processor)\ \} =$$

$$n * Ey(Dual\ Processor) \text{ ---------------------------- (6)}$$

$$n = \frac{Ex(Intel\ Single\ Processor) - Ex\ (Intel\ Dual\ Processor)}{Ey(Intel\ Dual\ Processor)} * \frac{100}{1}$$

-- (7)

$$n = \frac{11499 - 11119}{11119} * \frac{100}{1} \text{ ---------------------------- (8)}$$

$$n = \frac{380{*}100}{11119} = \frac{38000}{11119} \text{ ---------------------------- (9)}$$

$$n = 3.4176\% \text{ -------------------------------------- (10)}$$

Then,

$$SpeedUp = \frac{11499}{11119} = 1.0342 \text{ ------------ (11)}$$

The above results confirm that Intel Pentium Dual-Core Processor is over 3.42% faster than Intel Celeron Single Core Processor on SPEC CINT2006 benchmarks.

B. *Result of the Throughput of the Two Systems under Test*

Throughput is the total amount of work done in a given time i.e output relative to input; the amount passing through a system from input to output. SPECint_base2006 performs three runs of each benchmark in the test suite and records the median, so the boldly underlined text indicates a median measurement as shown in table 2 and 3. The median score with the higher ratio perform better, while the processor with the higher scores of the overall system performance of SPECint_base2006 shown in figure 3 and 4 are better. Table 5 shown the throughput performance of the two processors tested, the results show that Intel Pentium Dual-Core Processor is 3.11% faster than Intel Celeron Single Core Processor, while the throughput of Intel Pentium Dual-Core Processor is 1.03 times higher than Toshiba Satellite C660 (Intel Celeron Single Core Processor), figure 6 display the chart that clearly show the performance difference of the two processor.

	RESULT ON THE SYSTEM THROUGHPUT (SPECint_base2006)		
	SYSTEM	System Throughput SPECint_base_2006 Result	Overall Performance of Systems in percentage (%)
1	Intel Celeron Single Core	11.00	49.12
2	Intel Pentium Dual-Core Processor	11.40	50.89

Table 5 AMD Turion (tm) II P520 and Intel Pentium Dual Core Processor Throughput

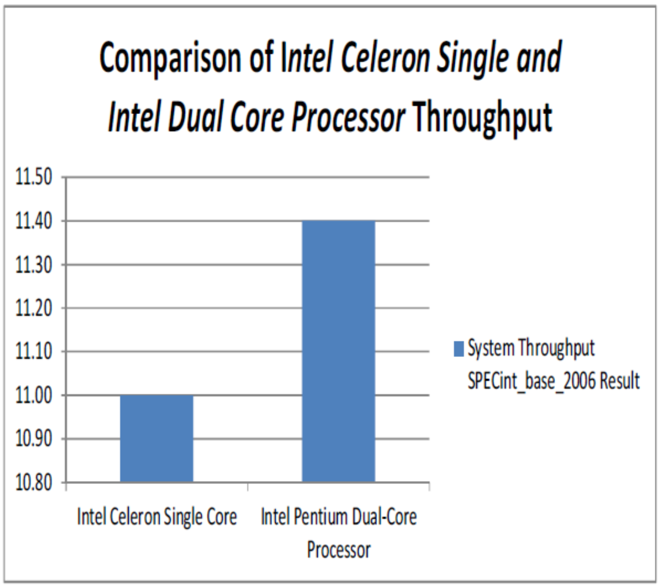


Figure 6: Comparison of Intel Celeron Single Core and Intel Pentium Dual Core Processor Throughput

6. Conclusion

The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used. The speedup that can be gained by using a particular feature is the factor of a parallel system as the ratio between the times taken by a single processor to solve a given problem to the time taken by a parallel system consisting of n processors to solve the same problem.
Intel Corporation has been leading in developing new technologies for the personal computer (PC). Intel single and dual core processors have been good for the consumer, resulting in constant innovation and lower prices; it gives computer users the opportunity to purchase computers in relation to their fields and cost of the system. This research measures the performance of Intel Celeron single core and Intel dual core processor, with the aids of SPEC CPU 2006 Benchmarks suite. The results showed that the execution time of Intel Pentium Dual-Core Processor is over 3.42% faster than Intel Celeron single core while the throughput of Intel Pentium Dual-Core Processor is 1.03 times higher than Intel Celeron single core. Intel Pentium Dual-Core Processor

had the best performance due to faster core-to-core communication, dynamic cache sharing between cores, smaller size of level 2 caches, and run at lower core and bus frequencies.

REFERENCE

[1]. O.S. Adeoye, **(2005);** Performance Evaluation of Pentium IV Processor, M.Tech Thesis, Federal University of Technology, Akure, Nigeria.

[2]. J. Emer, **(2005);** "Microprocessor Evolution:4004 to Pentium-4" Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

[3]. S.O. Falaki and O.S. Adewale, **(2000)**; Trends in Computer Systems Architecture, BJACS Volume 1, No 1, University of Benin Journal of Science.

[4]. A. I. Fasiku, **(2012);** Performance Evaluation of Multicore Processors, M.Tech Thesis, Federal University of Technology, Akure, Nigeria.

[5]. C. Hughes and T. Hughes, **(2008)**; "Professional Multicore Programming, (Design and Implementation for C++ Developers)", Published by Wiley Publishing, Inc. 10475 Crosspoint Boulevard, Indianapolis, IN 46256.

[6]. D. M. Pase and M. A. Eckl, **(2005)**; A Comparison of Single-Core and Dual-Core Opteron Processor Performance for HPC, IBM Systems and Technology Group, IBM xSeries Performance Development and Analysis, 3039 Cornwallis Road, Research Triangle Park, NC 27709-2195,USA.

[7]. D.A. Patterson and J.L. Hennessy, **(1996)**; 'Computer Architecture: A Quantitative Approach, 2nd Edition, Published by Morgan Kaufmann Publishers, inc. San Francisco California, Printed in the United States of America.

[8]. D.A. Patterson and J.L. Hennessy, **(2007)**; Computer Architecture A Quantitative Approach, 4th Edition, Published by Morgan Kaufmann publications, inc. San Francisco California, Printed in the United States of America.

[9]. M. K. Prabhu and K. Olukotun, **(2006)**; Exposing Speculative Thread Parallelism in SPEC2000, Stanford University, Computer Systems Laboratory, Stanford, California 94305.

[10]. SPEC, **(2006)**; Standard Performance Evaluation Corporation, www.spec.org, retrieved on 20/01/12.

[11]. G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel, **(2001)**; "The Microarchitecture of the Pentium® 4 Processor", Desktop Platforms Group, Intel Corp., Intel Technology Journal Q.