EXAMPLE ESSAY – Received 32 points = A grade

Computer Science Extended Essay

To what extent is simultaneous multithreading a viable means of increasing processor power?

Word Count: 3730

Candidate Number: xxxx

Centre Number: xxx

EXAMPLE ESSAY – Received 32 points = A grade

Abstract

The speed at which a computers central processing unit is able to complete tasks is both important to people using computers for everyday tasks, as it increases the usability of a system, but also in business environments where excess time spent performing calculations can cost a company money in milliseconds. Processor throughput is a measure of how many instructions a computer processor can perform in a given time period, and as such is a good measure of how fast a processor is.

There are many methods of increasing processor speeds, however I looked into how simultaneous multithreading was capable of increasing processor throughput. Moreover, I was interested in seeing how viable this technique is in increasing processor throughput in a variety of situations, by looking into its benefits and drawbacks. This is why I came to the research title: **"To what extent is simultaneous multithreading a viable means of increasing processor throughput?"**

In order to approach this question, I investigated the performance benefit of using a simultaneously multithreaded processor over an equivalent processor without it. This was done by running three benchmarking applications on the two processors and comparing the results. I then evaluated the results, researching possible reasons behind the difference in results between different tests, whereby I came to the conclusion that the workload of the processor has a huge impact on whether simultaneous multithreading has any real benefits.

After analysing the results from the experiment, I evaluated the viability of simultaneous multithreading as a method of increasing processor throughput, concluding that it is a viable means of increasing processor throughput, although there are some situations that it would be completely redundant.

[Word count: 274]

EXAMPLE ESSAY – Received 32 points = A grade

Table of Contents

EXAMPLE ESSAY – Received 32 points = A grade

To what extent is simultaneous multithreading a viable means of increasing processor throughput?

Introduction:

A computer's central processing unit (CPU) is an essential component within a computer system, and is responsible for managing the hardware and peripherals of a computer as well as processing and executing instructions.[1] Essentially overseeing the entirety of the operation of the computer means that this complex piece of circuitry must be immensely powerful, and as years have passed, one such way of making central processing units more powerful has been to increase the number of components on the circuit itself[2]. The logic behind this is simple, with more components on the circuit that makes up the CPU, the more there is available physically to be used within the CPU for processing and executing instructions[3]. In order to accommodate the growing number of components on the CPU[4], the size of these components has been consistently reduced to allow more and more of them to be packed into the same amount of space[5], which in turn has, in the past, sustained an increase in processor performance[6].

The need to explore more methods of increasing processor performance is largely fuelled by the ever growing desires of consumers to have faster, more efficient machines that are capable of doing more. However, relying on the shrinking size of components that make up CPUs is an unsustainable means

[1] Wingard, J. (2002). *Central Processing Unit Facts, information, pictures | Encyclopedia.com articles about Central Processing Unit*. [online] Encyclopedia.com. Available at: http://www.encyclopedia.com/topic/Central_Processing_Unit.aspx [Accessed 22 Feb. 2016].

[2] Intel, (2012). *Transistors to Transformations*. [online] Available at: http://www.intel.co.uk/content/dam/www/public/us/en/documents/corporate-information/museum-transistors-to-transformations-brochure.pdf [Accessed 22 Feb. 2016].

[3] Hruska, J. (2012). *The death of CPU scaling: From one core to many — and why we're still stuck | ExtremeTech*. [online] ExtremeTech. Available at: http://www.extremetech.com/computing/116561-the-death-of-cpu-scaling-from-one-core-to-many-and-why-were-still-stuck [Accessed 22 Feb. 2016].

[4] Ibid.

[5] Hruska, J. (2013). *Smaller isn't always better: The vanishing benefits (and profits) of smaller processes and new foundry tech | ExtremeTech*. [online] ExtremeTech. Available at: http://www.extremetech.com/computing/170130-smaller-isnt-always-better-the-vanishing-benefits-and-profits-of-smaller-processes-and-new-foundry-tech [Accessed 22 Feb. 2016].

[6] Mueller, S. (2006). *Microprocessors from 1971 to the Present | Microprocessor Types and Specifications | Que*. [online] Quepublishing.com. Available at: http://www.quepublishing.com/articles/article.aspx?p=482324&seqNum=2 [Accessed 22 Feb. 2016].

of increasing processor performance, as there will inevitably reach a point where components can physically no longer be reduced in size[7]. Other methods of increasing processor performance will begin to take precedence over shrinking manufacturing sizes in order to create better performing processors[8], and simultaneous multithreading is an example of this.

A way of measuring a processors performance is by measuring its throughput, throughput simply being how many units of information a computer system can process in a given time period[9]. This is why my essay will investigate the viability of simultaneous multithreading as a means of increasing processor throughput.

This essay will firstly investigate experimentally how much simultaneous multithreading can increase the throughput of a processor in order to evaluate its viability in the future as a means of increasing processor performance.

[7] Tally, S. (2012). *One and done: Single-atom transistor is end of Moore's Law; may be beginning of quantum computing*. [online] Purdue.edu. Available at: http://www.purdue.edu/newsroom/research/2012/120219KlimeckAtom.html [Accessed 22 Feb. 2016].

[8] Carrano, F. (2012). *The efficiency of algorithms*. [online] Pearson Education. Available at: http://www.spatial.cs.umn.edu/Courses/Spring12/1902/notes/Chapter04-EffeciencyOfAlgorithms.pdf [Accessed 22 Feb. 2016].

[9] Software.intel.com, (2008). *Measuring Instruction Latency and Throughput | Intel® Developer Zone*. [online] Available at: https://software.intel.com/en-us/articles/measuring-instruction-latency-and-throughput [Accessed 22 Feb. 2016].

Parallelism:

This essay will focus on simultaneous multithreading as a means of increasing processor throughput, which is a method of allowing applications to use thread level parallelism in order to increase the amount that a CPU can process.

Parallelism is a very broad term and in a non-computing context simply relates to something's ability to be broken down into smaller tasks that can be done simultaneously[10]. When applied in computing, this idea of performing multiple tasks simultaneously relates directly to a processor executing instructions[11]. Increasing parallelism in a processor would mean allowing more instructions to be executed simultaneously[12]. This would increase processor throughput, a measure of how many instructions a processor can execute in a given time period[13], by increasing the amount processed in a given time period.

When a computer executes a program, it is running a process[14]. Processes can contain one or more threads[15]. These threads provide the CPU with all the information it requires to execute a stream of instructions[16], and can be seen as a way to break down a process. You can describe processes as single threaded (only containing one thread)[17] and multi or highly threaded (containing more than one thread)[18]. On a processor that can handle only one thread at a time, it will seemingly do multiple

[10] Comer, D. (2006). *Computer Organisation*. [online] Cs.Purdue.com. Available at: http://www.eecs.wsu.edu/~hauser/teaching/Arch-F07/handouts/Chapter17.pdf [Accessed 22 Feb. 2016].
[11] Ibid.
[12] Ibid.
[13] Software.intel.com, (2008). *Measuring Instruction Latency and Throughput | Intel® Developer Zone*. [online] Available at: https://software.intel.com/en-us/articles/measuring-instruction-latency-and-throughput [Accessed 22 Feb. 2016].
[14] Msdn.microsoft.com, (2016). *Processes and Threads (Windows)*. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/ms684841(v=vs.85).aspx [Accessed 22 Feb. 2016].
[15] Ibid.
[16] Docs.oracle.com, (2015). *Processes and Threads (The Java™ Tutorials > Essential Classes > Concurrency)*. [online] Available at: https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html [Accessed 22 Feb. 2016].
[17] Merritt, T. (2012). *Multi-Threaded Application vs. Single Threaded Application - DZone Performance*. [online] dzone.com. Available at: https://dzone.com/articles/multi-threaded-application-vs [Accessed 22 Feb. 2016].
[18] Math.uni-hamburg.de, (2005). *Thread Scheduling*. [online] Available at: http://www.math.uni-hamburg.de/doc/java/tutorial/essential/threads/priority.html [Accessed 22 Feb. 2016].

things at once through scheduling when threads are allocated time to be executed by the processor and quickly switching between these different threads to make it appear as if multiple things are happening concurrently[19]. The processor will still, however, be only processing one thread at any one given time.

A simple example of parallelism that is implemented in many modern processors is the idea of multiple processor cores on a single processor chip. Each processor core is able to execute its own processes, and so multiple processes can be executed simultaneously by a multicore processor.[20]

Thread Level Parallelism:

Thread level parallelism extends parallelism into threads, a step further than parallelism in processes. Rather than having multiple processes running simultaneously on the different processor cores, thread level parallelism increases throughput by allowing multiple threads to be handled at the same time by a single processor core.[21]

The way a processor handles threads is by having an instruction pipeline. On a processor, each processor core has a pipeline which can handle threads[22]. The pipeline is a way of breaking down how a thread is handled within the CPU, which has multiple stages that are used to process the threads[23]. On a standard processor where there is nothing implemented to allow for thread level parallelism, one pipeline can only handle a single thread at a time.[24]

[19] Ibid.

[20] Software.intel.com, (2012). *Frequently Asked Questions: Intel® Multi-Core Processor Architecture | Intel® Developer Zone*. [online] Available at: https://software.intel.com/en-us/articles/frequently-asked-questions-intel-multi-core-processor-architecture [Accessed 22 Feb. 2016].

[21] Yalamanchili, S. (n.d.). *Thread level parallelism*. [online] Georgia Tech. Available at: http://users.ece.gatech.edu/~sudha/academic/class/ece3056/Lectures/8-Pipelined%20Datapath/pipelined-TLP.pdf [Accessed 22 Feb. 2016].

[22] Ibid.

[23] Hodgin, R. (2006). *What is a processor's pipeline? | Chips | Geek.com*. [online] @geekdotcom. Available at: http://www.geek.com/chips/what-is-a-processors-pipeline-561598/ [Accessed 22 Feb. 2016].

[24] Ibid.

Simultaneous Multithreading (SMT):

Simultaneous multithreading is a means of allowing thread level parallelism to be implemented. This works by not only allowing multiple threads to be handled by a single instruction pipeline, but also allows handling of threads concurrently within a pipeline.[25]

Because each pipeline is required to process multiple threads both at the same time and within the same pipeline, implementations of simultaneous multithreading require extra hardware within the CPU in order to compensate for the increased amount of threads that it will be handling[26]. Although extra hardware is required to implement SMT, it requires less hardware than adding an entire processor core to the processor. This is because on a simultaneously multithreaded processor, hardware to interpret threads is solely added, but not hardware that is capable of executing the thread (called the execution unit)[27], which is shared between threads.

On some operating systems such as Microsoft Windows 7, the extra hardware present in an SMT CPU is presented as extra processor cores by the operating system[28] to applications.

[25] Eggers, S., Emer, J., Leby, H., Lo, J., Stamm, R. and Tullsen, D. (1997). Simultaneous multithreading: a platform for next-generation processors. *IEEE Micro*, 17(5), pp.12-19.
[26] Ibid.
[27] Salunke, S. (2005). *Simultaneous Multithreading*. [online] University of Minessota. Available at: http://www.d.umn.edu/~salu0005/smt.pdf [Accessed 22 Feb. 2016].
[28] Support.microsoft.com, (2016). *Windows and Hyperthreading*. [online] Available at: https://support.microsoft.com/en-us/kb/810231 [Accessed 22 Feb. 2016].

Testing a simultaneously multithreaded processor:

In order to evaluate the viability of simultaneous multithreading in increasing processor throughput, I will first investigate how much of a performance improvement can be gained through using a simultaneously multithreaded processor over an equivalent processor without it.

In order to properly test the performance increase through using a SMT processor, the computer system used to test the performance difference will be kept identical: 8GB RAM, 500GB hard drive, NVidia GT 720 graphics processor, and the Intel i7-3770k central processing unit. The operating system being used is Windows 7. Controlling these variables will prevent variances in performance because of other aspects of the system and will make comparing results from the same processor more valid, due to a single variable (whether hyperthreading is enabled or disabled) changing at any one time.

The Intel i7-3770k CPU has been used as this processor implements Intel's hyperthreading (HT) technology, a form of simultaneous multithreading. Moreover, this hyperthreading technology can be enabled or disabled without changing any other aspect of the CPU, which makes results directly comparable between the processor when SMT is enabled and SMT is disabled, providing a clear indication of any performance benefits that SMT may cause. When the processor has hyperthreading enabled, it has 4 processor cores, each of which are simultaneously multithreaded to allow a total of 8 threads to be handled simultaneously. When the processor has hyperthreading disabled, it has 4 processor cores and the CPU is capable of handling 4 threads simultaneously.

Method of testing:

The processor will be tested using multiple benchmarking applications. These applications are designed specifically to test aspects of a computer system, in this case applications testing the CPU will be used.[29] These applications will then present a score for the system being tested, which reflects how well the system fared in the benchmark. These benchmarking applications will be run on the processor with hyperthreading enabled and disabled to allow for comparisons to be made.

The three benchmarking applications that will be used are: Cinebench, wPrime and SunSpider

Cinebench: This application uses the system's CPU to render a photorealistic image and measures how quickly this is done before converting this to a score. The multithreaded version of this benchmark splits the process of rendering the image into multiple threads, whereas the non-multithreaded version leaves the process of rendering the image in a single thread.[30]

wPrime: This application tests processor performance by getting the processor to compute complex mathematical equations recursively and measures the time taken to perform a set number of these calculations.[31]

SunSpider: This applications tests the ability of the CPU to process encryption and text manipulation, and measures the time taken to perform a predetermined set of manipulations as a measure of processor performance.[32]

[29] Rouse, M. (n.d.). *What is a benchmark? - Definition from WhatIs.com*. [online] SearchCIO. Available at: http://searchcio.techtarget.com/definition/benchmark [Accessed 22 Feb. 2016].
[30] Norris, J. (n.d.). *Cinebench*. [online] PCWorld. Available at: http://www.pcworld.com/article/238229/cinebench.html [Accessed 22 Feb. 2016].
[31] Wprime.net, (2012). *About wPrime | wPrime*. [online] Available at: http://www.wprime.net/About/ [Accessed 22 Feb. 2016].
[32] Webkit.org, (n.d.). *SunSpider 1.0.2 JavaScript Benchmark*. [online] Available at: https://webkit.org/perf/sunspider/sunspider.html [Accessed 22 Feb. 2016].

Hypothesis:

My hypothesis is that enabling hyperthreading, a form of simultaneous multithreading, will see a significant performance improvement in workloads that are highly threaded, and thus able to take advantage of higher thread level parallelism, however in workloads which aren't highly threaded there will be little or no improvement through having hyperthreading enabled.

Results:

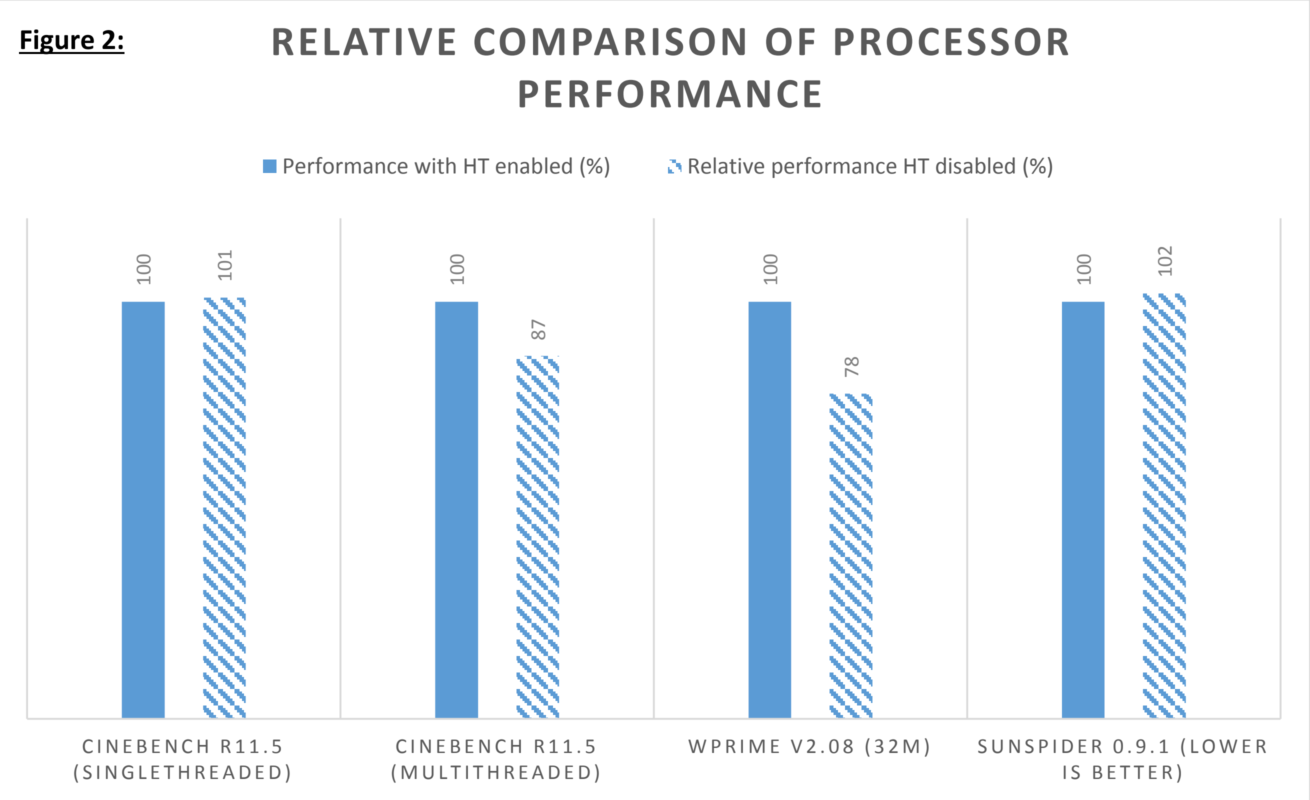
Figure 1 below shows the raw data taken from the different benchmarking applications of the processor with hyperthreading enabled and disabled:

Figure 1:

Benchmark Name	HT Enabled	HT Disabled	Relative % performance of HT disabled vs HT enabled
Cinebench R11.5 (Singlethreaded)	1.65	1.66	101
Cinebench R11.5 (multithreaded)	7.51	6.52	87
wPrime v2.08 (32M)	7.33	9.35	78
SunSpider 0.9.1 (lower is better)	122.30	119.70	102

Figure 2 below present the data from Figure 1 in the format of a graph:

Figure 2:



Conclusions of Testing:

Cinebench Single Threaded:

The single threaded CineBench benchmarking application showed no improvement in using hyperthreading over having it disabled. In fact, there was a slight performance degradation, although this was extremely small, just 1%, it was still a degradation in performance in having hyperthreading enabled. This is likely because of slight variances in the performance of the CPU.

Cinebench Multithreaded:

In the multithreaded variant of the CineBench benchmarking application, enabling hyperthreading was seen to significantly increase the throughput of the processor, the processor with hyperthreading disabled here performing 13% worse than the hyperthreading enabled counterpart.

Here it is clear that the added hardware in the CPU was able to be utilised in a very effective manner in order to increase the processor throughput. In this benchmark, now set to be highly threaded, the

CPU was able leverage the extra hardware available to process multiple threads simultaneously in order to increase the speed that the benchmark was completed, processing more in a smaller amount of time and increasing the processor's throughput.

In the context of rendering an image, which is what the benchmark application has the CPU do, it is clear how SMT would be of great advantage. Rendering each pixel on the screen could be split up into multiple independent threads, which through using SMT to process each thread simultaneously would increase processor throughput dramatically.

WPrime:

In the WPrime benchmarking application, enabling hyperthreading once more saw a significant performance increase over disabling hyperthreading, whereby the hyperthreading disabled performed 22% worse than the hyperthreading enabled equivalent.

Once more the extra hardware available for use within the CPU to allow for hyperthreading allowed the processor to perform more calculations simultaneously and thus increased the processor throughput greatly. This presents how in a context of performing heavy mathematical calculations on a CPU SMT could be extremely beneficial, whereby the task of performing multiple calculations can be split up into separate threads and executed more quickly on the CPU because SMT.

SunSpider:

In the SunSpider benchmarking application, there was seen to be performance degradation from enabling hyperthreading, whereby the hyperthreaded processor performed 2% worse than the non hyperthreaded equivalent.

Once again this is an unexpected result that could be caused due to variance in the performance of a CPU, however this may not be the case.

Performance degradation from enabling hyperthreading:

Although the performance degradation from enabling hyperthreading in my tests was seen to be small, and possibly due to variance, there are many reasons why there might be a performance decreased when enabling hyperthreading, a form of SMT.

Because SMT requires extra hardware within the CPU to handle extra threads, but also shares some resources between threads being processed concurrently, performance may have degraded, for example in the SunSpider benchmark, because threads that were being concurrently may have required the same part of the CPU at the same time, for example the execution unit, which is shared between threads. This would cause one of the threads to stall, simply meaning the thread is unable to run[33]. This would cause the processor throughput to be reduced as threads are having to wait before being executed, reducing the amount that the processor can process in a given time period.

Furthermore, on the Windows 7 operating system, the extra hardware on a SMT processor are presented as extra processor cores to applications. On a CPU with multiple processor cores, much like the processor used in my investigation, each capable of handling threads, this could cause unnecessary stalling, as there may be other full processor cores free to execute the thread which aren't being used whilst the extra hardware is instead being utilised to process threads.[34] This would force threads to share hardware which could slow down the processing of the threads being executed concurrently.

Overall conclusions of results:

The results clearly show that in applications capable of utilising the extra hardware within the CPU, hyperthreading can cause a significant improvement in processor throughput. Although this is the case, there are situations where hyperthreading didn't cause any speed up in performance, namely

[33] Harned, E. (2014). *Detecting and handling stalled threads in JavaSE*. [online] Coopsoft.com. Available at: http://coopsoft.com/ar/StalledArticle.html [Accessed 22 Feb. 2016].

[34] Bitsum.com, (2015). *When HyperThreading Hurts*. [online] Available at: https://bitsum.com/pl_when_hyperthreading_hurts.php [Accessed 22 Feb. 2016].

the single threaded benchmark, and in fact there was also a reduction in performance due to enabling hyperthreading.

Overall however, the improvements seen by implementing simultaneous multithreading on a process depends greatly on the workload the processor is processing, which supports my hypothesis, whereby the benchmarking applications that were more heavily threaded saw a large performance improvement though enabling hyperthreading.

Evaluating the testing methods:

The benchmarking applications used were appropriate in highlighting the clear benefits of implementing a form of simultaneous multithreading within a processor, and allowed comparisons between a SMT and a non SMT to be formed. Furthermore, the controlling of all other aspects of the system meant that results were valid and comparable.

Having completed this experiment, another relevant experiment that would be interesting to investigate would be to see how SMT scales with a processor with more or less cores. Clearly my experiment revealed that SMT improves performance on certain workloads by a significant amount (over 10% on two of the benchmarks), however this improvement may be larger or smaller depending on the number of full cores the processor has. The test system used a CPU that has 4 processor cores, each able to handle up to two threads depending on whether hyperthreading was enabled or not. If a CPU had, for example, only two cores each able to handle two threads with hyperthreading enabled, the performance improvement may have been greater, as the reduced number of full cores available would have left less capacity within the CPU for threads to be processed concurrently, which would mean that the relative effect of enabling hyperthreading would be greater.

Although the benchmarking applications were suitable in highlighting the overall performance benefit of SMT, more specific programs and workloads could have been tested to find out what specifically SMT would work well with and in what applications there would be little to no improvement through using SMT. However, the suite of applications used did present the general trend showing more threaded applications to benefit from SMT, and non-multithreaded applications seeing no improvement, and in our tests a performance decrease for these single threaded applications.

Testing the performance of a SMT processor on different operating systems could also possibly yield different results. The way the operating system handles the extra hardware required to allow for simultaneous multithreading depends on the operating system, and so on operating systems other than Windows 7 other results may have been obtained. In Windows 7, because the extra hardware are presented as extra processor cores on the CPU, this allows for possible degradation in performance of the processor as previously mentioned. This may not be the case on other operating systems, which could handle the extra hardware differently.

Furthermore, the experiment used only tested Intel's hyperthreading, which is a specific implementation of SMT within a processor. In order to further strengthen arguments made using our results, other processors with different implementations of SMT could be tested.

Evaluating the viability of simultaneous multithreading as a means of increasing processor throughput:

Having completed the investigation, it is clear that simultaneous multithreading, in some circumstances is a valid and effective method of increasing processor throughput. However, the extent to which this is true depends greatly on the workload of the processor. This performance improvement relies greatly on software and applications that are able to be split into multiple threads to be processed, increasing thread level parallelism within the processor.

In the future where shrinking the components on a processor in order to increase its performance and throughput will become increasingly difficult, methods of increasing processor throughput will need to shift from hardware based solutions to software based solutions. In this case, applications could become more and more multithreaded in order to allow the application to run as well as possible on machines which are able to handle more threads. With this likely being the case, SMT would see a greater performance improvement in a wider range of applications.

The benefits of SMT are clearly evident in applications that can utilise its capabilities. As the CineBench benchmark presented, in rendering an image, SMT can greatly increase the time taken to execute the process. For this reason, many graphical processing units, processors designed specifically for rendering images and graphics[35], are simultaneously multithreaded and are capable of handling hundreds of threads simultaneously[36]. This supports simultaneous multithreading as a viable means of increasing processor throughput, as its effectiveness in increasing throughput is so great that it is already being used in processors designed for specific purposes that would benefit from SMT.

Although SMT requires extra hardware on the CPU in order to be implemented, the extra hardware required is less than that required to add more processor cores to a processor. For this reason, it

[35] Renderingpipeline.com, (2012). *Understanding the parallelism of GPUs | RenderingPipeline*. [online] Available at: http://renderingpipeline.com/2012/11/understanding-the-parallelism-of-gpus/ [Accessed 22 Feb. 2016].
[36] Ibid.

makes SMT more commercially viable for manufacturers of processors, as it is a means of increasing the throughput of a processor without having to spend as much as adding more cores to the processor. This can be reflected by Intel's decision to implement their hyperthreading technology on many of their processors[37], from low end consumer grade processors to high end processors used within servers. The support of this technique in increasing processor throughput by such a large manufacturer of computer processors is a testament to its viability in the future for increasing processor throughput.

The need for extra hardware to support SMT, however, could be seen as detrimental in its viability in the future for smaller power efficient processors in the future. The extra hardware required to implement SMT means that there is a greater power draw and increased heat output from processor that implement this technology[38]. Because of this, it might not be a viable means of increasing throughput on systems that have limited cooling abilities, on smartphones for example where often there isn't any active cooling solution at all, and the processor must be cooled passively.

Furthermore, as SMT relies on sharing resources on a CPU, for some applications, SMT will never be a viable means of increasing processor throughput. Depending on what exactly is shared within the CPU, there will always be times when the shared resources in the CPU are required by two threads that are being processed concurrently. Because of this, if there are applications which will always have this type of workload, SMT would be rendered completely useless and could even cause slowdown within the application. For this reason, SMT is not viable in applications where shared resources, such as the execution unit, would be required for most threads[39], as it would cause continued stalling of threads, slowing down the execution of the task.

[37] Intel® ARK (Product Specs), (2016). *ARK | Processor Feature Filter*. [online] Available at: http://ark.intel.com/search/advanced?HyperThreading=true [Accessed 22 Feb. 2016].

[38] Henry, A. (2013). [online] Lifehacker.com. Available at: http://lifehacker.com/how-hyper-threading-really-works-and-when-its-actuall-1394216262 [Accessed 22 Feb. 2016].

[39] Bitsum.com, (2015). *When HyperThreading Hurts*. [online] Available at: https://bitsum.com/pl_when_hyperthreading_hurts.php [Accessed 22 Feb. 2016].

However, as the situations where the processor will need to handle threads that all need the same shared resource on an SMT processor would likely be very specific, the overall performance improvement in other applications that can be gained through implementing SMT present it as a viable and effective means of increasing processor throughput in the future.

Conclusion:

Having completed the experiment, I was able to evaluate the extent to which simultaneous multithreading is a viable means of increasing processor throughput. I found that SMT was able to significantly improve the throughput of a processor in multithreaded applications, increasing performance by over 10% in the two benchmarking applications tested. In the single threaded benchmarking applications, there was no performance gain through using simultaneous multithreading.

Completing the investigation also brought up the question of how SMT scales with different processors with more or less processor cores than the system that I tested, and would be an interesting way of furthering the investigation in the future, and also how results would vary based on the operating system tested due to differences in the way SMT is handled by operating systems.

Combining the results from my experiment with research, I reached the conclusion that simultaneous multithreading is a viable means of increasing processor throughput, and will continue to be a viable means in the future. Because simultaneous multithreading offers a performance improvement in multithreaded applications, and it is likely that applications will become more multithreaded in the future, this supports SMT as a viable means of increasing processor throughput, and the benefits of implementing SMT likely becoming more substantial in the future in a wider range of applications.

Although the performance gains from implementing SMT on a processor vary based on workload, where in my experiment two benchmarking applications showed there was no improvement to using

SMT and theoretically there could be situations where SMT slows down system performance, the possible increases in performance through using SMT can be substantial. SMT is currently, and will continue in the future to be a viable means of increasing processor throughput for multithreaded applications, but is not appropriate for all processors, as some workloads will never see a benefit to implementing SMT.

[Word Count : 3730]

Bibliography:

Bitsum.com, (2015). *When HyperThreading Hurts*. [online] Available at: https://bitsum.com/pl_when_hyperthreading_hurts.php [Accessed 22 Feb. 2016].

Carrano, F. (2012). *The efficiency of algorithms*. [online] Pearson Education. Available at: http://www.spatial.cs.umn.edu/Courses/Spring12/1902/notes/Chapter04-EffeciencyOfAlgorithms.pdf [Accessed 22 Feb. 2016].

Comer, D. (2006). *Computer Organisation*. [online] Cs.Purdue.com. Available at: http://www.eecs.wsu.edu/~hauser/teaching/Arch-F07/handouts/Chapter17.pdf [Accessed 22 Feb. 2016].

Docs.oracle.com, (2015). *Processes and Threads (The Java™ Tutorials > Essential Classes > Concurrency)*. [online] Available at: https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html [Accessed 22 Feb. 2016].

Eggers, S., Emer, J., Leby, H., Lo, J., Stamm, R. and Tullsen, D. (1997). Simultaneous multithreading: a platform for next-generation processors. *IEEE Micro*, 17(5), pp.12-19.

Harned, E. (2014). *Detecting and handling stalled threads in JavaSE*. [online] Coopsoft.com. Available at: http://coopsoft.com/ar/StalledArticle.html [Accessed 22 Feb. 2016].

Henry, A. (2013). [online] Lifehacker.com. Available at: http://lifehacker.com/how-hyper-threading-really-works-and-when-its-actuall-1394216262 [Accessed 22 Feb. 2016].

Hodgin, R. (2006). *What is a processor's pipeline? | Chips | Geek.com*. [online] @geekdotcom. Available at: http://www.geek.com/chips/what-is-a-processors-pipeline-561598/ [Accessed 22 Feb. 2016].

Hruska, J. (2012). *The death of CPU scaling: From one core to many — and why we're still stuck | ExtremeTech*. [online] ExtremeTech. Available at: http://www.extremetech.com/computing/116561-the-death-of-cpu-scaling-from-one-core-to-many-and-why-were-still-stuck [Accessed 22 Feb. 2016].

Hruska, J. (2013). *Smaller isn't always better: The vanishing benefits (and profits) of smaller processes and new foundry tech | ExtremeTech*. [online] ExtremeTech. Available at: http://www.extremetech.com/computing/170130-smaller-isnt-always-better-the-vanishing-benefits-and-profits-of-smaller-processes-and-new-foundry-tech [Accessed 22 Feb. 2016].

Intel® ARK (Product Specs), (2016). *ARK | Processor Feature Filter*. [online] Available at: http://ark.intel.com/search/advanced?HyperThreading=true [Accessed 22 Feb. 2016].

Math.uni-hamburg.de, (2005). *Thread Scheduling*. [online] Available at: http://www.math.uni-hamburg.de/doc/java/tutorial/essential/threads/priority.html [Accessed 22 Feb. 2016].

Merritt, T. (2012). *Multi-Threaded Application vs. Single Threaded Application - DZone Performance*. [online] dzone.com. Available at: https://dzone.com/articles/multi-threaded-application-vs [Accessed 22 Feb. 2016].

Msdn.microsoft.com, (2016). *Processes and Threads (Windows)*. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/ms684841(v=vs.85).aspx [Accessed 22 Feb. 2016].

Mueller, S. (2006). *Microprocessors from 1971 to the Present | Microprocessor Types and Specifications | Que*. [online] Quepublishing.com. Available at: http://www.quepublishing.com/articles/article.aspx?p=482324&seqNum=2 [Accessed 22 Feb. 2016].

Norris, J. (n.d.). *Cinebench*. [online] PCWorld. Available at: http://www.pcworld.com/article/238229/cinebench.html [Accessed 22 Feb. 2016].

Renderingpipeline.com, (2012). *Understanding the parallelism of GPUs | RenderingPipeline*. [online] Available at: http://renderingpipeline.com/2012/11/understanding-the-parallelism-of-gpus/ [Accessed 22 Feb. 2016].

Rouse, M. (n.d.). *What is benchmark? - Definition from WhatIs.com*. [online] SearchCIO. Available at: http://searchcio.techtarget.com/definition/benchmark [Accessed 22 Feb. 2016].

Salunke, S. (2005). *Simultaneous Multithreading*. [online] University of Minessota. Available at: http://www.d.umn.edu/~salu0005/smt.pdf [Accessed 22 Feb. 2016].

Software.intel.com, (2008). *Measuring Instruction Latency and Throughput | Intel® Developer Zone*. [online] Available at: https://software.intel.com/en-us/articles/measuring-instruction-latency-and-throughput [Accessed 22 Feb. 2016].

Software.intel.com, (2012). *Frequently Asked Questions: Intel® Multi-Core Processor Architecture | Intel® Developer Zone*. [online] Available at: https://software.intel.com/en-us/articles/frequently-asked-questions-intel-multi-core-processor-architecture [Accessed 22 Feb. 2016].

Software.Intel.com, (2012). *Transistors to Transformations*. [online] Available at: http://www.intel.co.uk/content/dam/www/public/us/en/documents/corporate-information/museum-transistors-to-transformations-brochure.pdf [Accessed 22 Feb. 2016].

Support.microsoft.com, (2016). *Windows and Hyperthreading*. [online] Available at: https://support.microsoft.com/en-us/kb/810231 [Accessed 22 Feb. 2016].

Tally, S. (2012). *One and done: Single-atom transistor is end of Moore's Law; may be beginning of quantum computing*. [online] Purdue.edu. Available at: http://www.purdue.edu/newsroom/research/2012/120219KlimeckAtom.html [Accessed 22 Feb. 2016].

Webkit.org, (n.d.). *SunSpider 1.0.2 JavaScript Benchmark*. [online] Available at: https://webkit.org/perf/sunspider/sunspider.html [Accessed 22 Feb. 2016].

Wingard, J. (2002). *Central Processing Unit Facts, information, pictures | Encyclopedia.com articles about Central Processing Unit*. [online] Encyclopedia.com. Available at: http://www.encyclopedia.com/topic/Central_Processing_Unit.aspx [Accessed 22 Feb. 2016].

Wprime.net, (2012). *About wPrime | wPrime*. [online] Available at: http://www.wprime.net/About/ [Accessed 22 Feb. 2016].

Yalamanchili, S. (n.d.). *Thread level parallelism*. [online] Georgia Tech. Available at: http://users.ece.gatech.edu/~sudha/academic/class/ece3056/Lectures/8-Pipelined%20Datapath/pipelined-TLP.pdf [Accessed 22 Feb. 2016].