Enhancing Cryptographic Performance on Raspberry Pi 4: Analyzing Cipher Techniques and Miller-Rabin

Suhail Al Marzooqi

Master of Science
Cyber Security, Privacy and Trust
School of Informatics
University of Edinburgh
2023

Abstract

This research investigates the performance optimization of cryptographic operations on a Raspberry Pi 4 system, encompassing several AES cryptographic modes, including Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Miller-Rabin primality test. By employing Transparent Huge Pages (THP), multithreading, OpenMP, and specific software optimization techniques, we aimed to leverage the system's hardware capabilities fully. The findings illustrate significant performance boosts with the application of THP, particularly in inherently parallelizable modes such as ECB. Furthermore, Miller-Rabin, in one of the implemented methods, stood out for its superior branch prediction rates and reduction in branch misses. On the other hand, the other method, despite having a higher count of total branch predictions, demonstrated enhanced flow control through adept multithreading. It is imperative to note the potential increased memory demands in CBC and CFB due to pipelining. This investigation not only emphasizes the capabilities of the Raspberry Pi 4 for cryptographic operations but also suggests broader applicability for its optimization techniques, providing a pathway for future innovation in both academic and industrial settings.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics Committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Suhail Al Marzooqi*)

Acknowledgements

First and foremost, I would like to thank my supervisor, Yuvraj Patel, for his continuous support of this dissertation and for always pushing us to do our best, no matter the challenge.

I would also love to thank my wife, Shamma, for always being there for me whenever I needed her, as we are both doing our dissertations.

Following that, I would like to thank my family members; My dad for his motivational speeches, my mom for the funny talks, and my sister and two brothers for believing in me always.

I would also like to take this chance to thank the sponsor of my Masters, Technology Innovation Institute; mainly, I would like to thank the leadership for their trust in us. Our Chief Shreekant "Ticky" Thakkar for his continuous support and pushing us always to reach the sky and higher. Jean-Pierre Giacalone, Martin Andreoni for guiding me with my career and always pushing me to do my absolute best. Michael Baddeley for his support and the one who pushed me to join Edinburgh University, the best Scottish guy I know. For their unwavering support, expert guidance, and encouragement, Willian Lunardi and Govind Singh have been invaluable throughout this process. and not only for being exceptional mentors but also for bringing joy and laughter into our interactions. A special thank you to Oksana Sandulenko, the warrior behind the scenes; she was there whenever we needed her! Finally, I want to thank the HR team and everyone I did not mention for their invaluable support.

Finally, I want to thank me for believing in me; I want to thank me for doing all this hard work. I want to thank me for having no days off. I want to thank me for never quitting. I want to thank me for always being a giver and trying to give more than I receive. I want to thank me for trying to do more right than wrong!

Table of Contents

Chapter 1

Introduction

1.1 Project Motivation

For decades, servers have served as vital nerve centers within IT infrastructures, bestowing businesses, and organizations with robust computing resources essential for critical tasks. However, these conventional servers, though reliable, are burdened by substantial price tags and considerable operating expenses. Notably, issues arise concerning power consumption and cooling requisites. As an alternative, the Raspberry Pi is a cost-efficient and energy-conserving option. The Raspberry Pi's popularity has soared meteorically due to its affordability, versatility, and energy efficiency. Its economical and energy-saving attributes have found utility in a myriad of applications. From smart home automation and media streaming to educational endeavors and basic computing tasks, the Raspberry Pi has carved its niche. Yet, it comes with inherent limitations. The Raspberry Pi's restricted processing power, memory capacity, storage, and network capabilities have often rendered it inadequate for demanding high-performance computing needs.

Nevertheless, recent initiatives have orchestrated Raspberry Pi clusters to mimic high-performance computing systems, positioning them as potential rivals to traditional servers [1, 9]. These studies have predominantly focused on specific workloads, pitting their performance against that of conventional servers. However, a critical challenge lies in identifying workloads that can optimally function within a Raspberry Pi cluster, exploiting its architecture to its fullest. Prior investigations were hindered by the limitations of older Raspberry Pi versions, constrained by their modest processing power, memory, storage, and networking capabilities. Notably, the Raspberry Pi 4 Model B has made significant strides in these domains, augmenting its viability as a

computing platform. By delving deep into the operational nuances of workloads on the Raspberry Pi, it could eventually evolve into a credible alternative to traditional servers.

1.2 Objectives

The central focus of this project lies in exploring the viability of utilizing a Raspberry Pi 4 Model B [5] to handle computationally-intensive workloads in comparison to the performance of traditional servers. The primary objective is to identify workloads that can effectively function and comprehend computations, optimizing the utilization of Raspberry Pi's resources at a profound low level. The insights gleaned from this study have the potential to unlock novel avenues for Raspberry Pi applications, offering cost-effective and energy-efficient solutions for high-performance computing. The following are the key goals of this project:

- Select an intensive workload and gain an in-depth understanding of its operation
- Demonstrate the efficiency of the workload with different events running on the Raspberry Pi
- Optimize the workloads and demonstrate the efficiency of the Raspberry Pi
- Analyze the workload and its overall efficiency

1.3 Structure

- Background: Explanation of technical concepts essential for comprehending the paper
- Related Work: Exploration of papers that have influenced and inspired this project
- Methodology: Description of the experimental setup, the hardware used, and software configurations employed
- Implementation and Evaluation: Analysis of the experiments and comparisons to original codes
- Discussion and Conclusion: The findings of this study shed light on several important aspects, encompassing strengths, weaknesses, and potential directions for future research

Chapter 2

Background

In this chapter, we provide the necessary background information for the reader to understand the thesis. Sec. 2.1 provides an overview of Raspberry Pi and its components. This chapter wraps up in Sec. 2.6. Here, we talk about past studies that looked at clusters of Raspberry Pi or specific tasks. We also highlight what makes our project stand out, especially when choosing the right cryptographic tasks.

2.1 Raspberry Pi

The following is based on my IPP report. The Raspberry Pi is a series of compact single-board computers that originated in the United Kingdom by the Raspberry Pi Foundation. The main objective behind this project was to foster the learning of fundamental computer science in schools and impoverished nations. This tiny, affordable device can be programmed to execute many tasks [13].

The inaugural Raspberry Pi model was launched in February 2012. Since then, there have been multiple iterations of the device, with the Raspberry Pi 4 Model B being the most recent version. Raspberry Pi, a diminutive and budget-friendly single-board computer, comprises several components, each serving a specific function. The core components of a Raspberry Pi encompass [10]:

- **System on a Chip (SoC):** Acting as the main processing unit, the SoC integrates the CPU, GPU, and other crucial components.
- RAM: The Raspberry Pi's memory is used to temporarily store data and program code while the device is operational.

- **GPIO Pins:** The General Purpose Input/Output pins facilitate the connection of external devices such as sensors, motors, and other electronic components.
- **USB Ports:** These link external USB devices like keyboards, mice, and storage peripherals to the Raspberry Pi.
- **Ethernet Port:** An Ethernet port enables the connection of the Raspberry Pi to a network via an Ethernet cable.
- **Wi-Fi Module:** The Raspberry Pi can also access a network wirelessly using its in-built Wi-Fi module.
- HDMI Port: The HDMI port connects the Raspberry Pi to a display device like a monitor or TV.
- MicroSD Card Slot: This slot is designed to house a MicroSD card that contains the operating system and other essential files needed for the Raspberry Pi's functioning.
- Power Port: The power port connects to a power source, a USB charger, or a dedicated power supply unit.

In its early iterations, the Raspberry Pi fell short of matching an average desktop PC's performance in all respects. Desktop PCs were considerably more advanced, and this continued for several years. However, this dynamic has shifted with the advent of the Raspberry Pi 3 and, more prominently, the Raspberry Pi 4. The Raspberry Pi 4, with its augmented power, can now compete with and even outpace an average desktop PC under certain circumstances [8]. This progression has positioned the Raspberry Pi 4 as a plausible alternative to a primary entry-level desktop computer.

2.2 AES

The Advanced Encryption Standard (AES) [14], inspired by the Rijndael algorithm [3], showcases the cryptographic expertise of Joan Daemen and Vincent Rijmen. This symmetric block cipher uses the same keys for both encryption and decryption and is designed for 128-bit data blocks. The encryption keys can be 128, 192, or 256 bits. Using a structured procedure, AES transforms plaintext into ciphertext through multiple rounds of specific transformations. The key's length determines the number of rounds: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Examining the AES process more closely:

- **SubBytes:** This initial step substitutes state data using a carefully constructed S-box table resistant to common cryptanalytic attacks.
- **ShiftRows:** This phase cyclically shifts data rows to the left, each with a unique magnitude, ensuring byte permutations across columns.
- **MixColumns:** Here, each column is viewed as a polynomial over a particular finite field. It undergoes a modulo multiplication with an irreducible polynomial, blending bytes in every column.
- **AddRoundKey:** The process ends by combining the round key with the state through a bitwise XOR. This round key is derived from the main encryption key using a specific key scheduling method.

Expanding on AES, various operational modes like ECB, CBC, and CFB have been developed.

- **ECB** (depicted in Figure G.1): This is the simplest mode, encrypting each plaintext block independently. While easy and highly parallelizable, it is susceptible to pattern-based attacks.
- **CBC** (presented in Figure G.2): Introduces a chaining concept. Every plaintext block is XORed with the previous ciphertext block before encryption, creating dependencies. An initialization vector (IV) enhances security. This chaining hides patterns well but does restrict parallelization.
- **CFB** (highlighted in Figure G.3): This mode transforms AES into a stream cipher using a feedback mechanism to create a keystream. XORing this with the plaintext produces the ciphertext. However, a singular error in the ciphertext can affect the decryption of the following blocks. (CFB has been omitted from the thesis but the optimizations are as such for the CBC)

Over the past twenty years, AES has endured extensive cryptographic scrutiny and is a trusted encryption standard. Its wide-ranging uses cover secure data communication and protected data storage. Its blend of a straightforward design and unmatched encryption strength cements AES's role as the preferred encryption solution for many applications.

2.3 Transparent Huge Pages

Transparent Huge Pages (THP) [2] is a technique within the Linux kernel that boosts system performance by automatically handling ”huge pages.” Before diving into THP, it is crucial to grasp the roles of pages and page tables in Linux’s memory management.

In Linux, a ”page” is the tiniest memory chunk the kernel’s memory manager works with. These pages are organized into structures called page tables. With the usual 4KB pages, page tables can grow quite large in systems with ample RAM, increasing memory, and CPU use. To address this, Linux introduced ”huge pages” much larger than the standard 4KB. By employing these, the page table entries are considerably reduced, cutting down on the overhead. One crucial advantage of using huge pages is the reduction of page faults. Page faults occur when a program attempts to access a page in memory that isn’t currently present in the system’s RAM. By using larger pages, the system can store more data contiguously, thus reducing the chances of a page fault. Moreover, the Translation Lookaside Buffer (TLB), becomes more effective with THP. Since the TLB can cache the address translations of large memory regions, huge pages ensure that the TLB covers a larger memory address range with fewer entries, leading to fewer TLB misses and, thus improved performance.

However, managing huge pages manually can be tricky. That is where THP steps in. It is a Linux kernel feature that handles huge pages automatically without changes to application codes. When activated, THP assigns huge pages to suitable applications and reverts to regular pages when needed. All this happens behind the scenes from the application’s standpoint—hence the ”transparent” in its name. Moreover, THP can split a huge page back into regular ones if necessary. It can also merge existing regular pages into a huge page during operation, potentially optimizing performance. Still, THP is not a universal fix. While it can boost performance in some situations, it might hinder it in others. So, it is essential to test THP exhaustively before implementing it widely. In essence, THP in the Linux kernel is a sophisticated memory management tool designed to enhance performance, minimizing manual oversight and avoiding changes to application codes.

2.4 Miller-Rabin Primality Test

The Miller-Rabin test is a probabilistic algorithm used to determine whether a given number is a prime number. It was developed by Gary L. Miller in 1976 and later

improved by Michael O. Rabin [12]. The test relies on the property that if a number n is a prime number, it has specific properties regarding its factors. The Miller-Rabin algorithm randomly selects a witness a and checks whether the properties hold for n. If n passes these checks for several random witnesses a, then it is considered "probably prime" with a high degree of confidence. The algorithm works as follows:

1. Given an odd number n to test for primality and find k and q such that $n-1 = 2^k * q$, where q is an odd number.
2. Randomly select a witness a from the range [2, n-2]
3. Compute $a^q mod n$ and check if the result is congruent to 1 or -1 (mod n).
4. If the result is congruent to 1 or -1, then n is probably prime for this witness a.
5. Repeat steps 2-4 for multiple witnesses to increase confidence in the result.

The Miller-Rabin test can produce false positives, possibly classifying a composite number as prime. However, using multiple witnesses and certain predefined values can significantly reduce the probability of false positives, making it a widely used and effective primality testing algorithm.

One important thing to point out is in Chapter 4. We will be using two distinct methods to test out the Miller-Rabin Primality test. Mainly, we refer to them as methods 1 and 2. Regarding their innate differences, Method 1 favors prime number generation by manually creating a number with a defined bit size and then adjusting it to be odd. In contrast, Method 2 generates two random numbers, merges them using a bitwise OR operation, and subsequently ensures the result is odd. The consequence is a more extensive number distribution in Method 2, which possibly boosts the number of numbers to check but reduces the complexity of the number generation. In conclusion, both methods share the time complexity of the Miller-Rabin test, which stands at $O(k \log n \log n)$. The decision between methods and optimization levels is intricately tied to the task's nature. The same primality test is used, but the random number generation differs. We will use Method 1 to generate 8 64-bit prime numbers, while in Method 2, we will use it to generate at least 5000 prime numbers.

2.5 Related Work

This is based on my IPP report. Numerous studies have highlighted the Raspberry Pi's potential in diverse computational contexts: Hajji and Tso [9] showcased its

promise for smaller big data workloads but with performance declines for larger sets; Mados et al. [11] revealed its remarkable efficiency as a web server against traditional servers, necessitating a broader evaluation; while Dubey and Kagdi [4] demonstrated its effectiveness in matrix multiplication within cluster configurations, signaling the need for diversified workload assessments.

Mados et al. [11], in their unique study, contrasted the Raspberry Pi 3 with a traditional 1U Intel Rack server, assessing energy efficiency and HTTP request response times. Notably, the Raspberry Pi 3 displayed impressive efficiency, handling 10,000 concurrent HTTP requests with a mere 23% average processor load, while the rack server reached up to 87%. This positioned the Raspberry Pi 3 as a compelling choice for compact web server design, although it necessitates a broader evaluation considering factors like scalability and data security.

Meanwhile, Dubey and Kagdi [4] focused on the efficiency of Raspberry Pi cluster supercomputers for matrix multiplication. Their research showed a direct correlation between the number of Raspberry Pis in a cluster and time efficiency. The more units in the cluster, the quicker the matrix multiplication tasks concluded, suggesting the importance of workload distribution and reduced communication latency. Yet, their study's confined scope to matrix multiplication signals a need for a more diversified evaluation of Raspberry Pi clusters' performance.

In summary, these studies underline the vast potential of Raspberry Pi in various computational scenarios. They also collectively highlight areas that require further exploration to optimize performance and functionality. Nevertheless, the previous studies have centered on using Raspberry Pis for advanced computing, often picking specific tasks and measuring their efficiency against standard servers. However, our research diverges, aiming to pinpoint tasks that best fit a network of Raspberry Pis, considering their distinct structure, constraints, and strengths as cryptographic operations are demanding regarding resources.

Chapter 3

Methodology

3.1 Workload Selection

Our experimental methodology utilizes the Raspberry Pi 4, a compact computing powerhouse powered by the Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC, complemented by a RAM option of 8GB, dependent on the chosen model. The system runs on the Ubuntu 22.04 operating platform.

A standout characteristic of the Raspberry Pi 4, central to our investigation, is its bifurcated CPU cache arrangement: L1 and L2. The primary L1 cache has distinct allocations for instructions (L1i) and data (L1d). In contrast, the L2 cache acts as the backup cache tier. The importance of optimizing Raspberry Pi arrays lies in the precise task selection, harmonizing with its distinctive architecture and features. It is critical to align tasks with the Pi's specific cache dimensions, notably the L1 and L2 layers. Aligning tasks with these cache capacities can substantially minimize cache misses, thus boosting performance. Given the Pi's modest memory allocation in comparison to mainstream servers, it becomes crucial to choose tasks that operate within these bounds. Tasks that heavily tax memory can introduce inefficiencies, such as page faults. Furthermore, despite the Raspberry Pi's competent networking features, its bandwidth and latency deviate from regular server metrics. So, tasks that lean heavily on expansive network data exchange might need tailored adjustments to capitalize on Pi's network features. Essentially, astute task allocation, cognizant of cache, memory, and network constraints, is indispensable for achieving peak Raspberry Pi array performance. Such a concentrated strategy ensures we harness Raspberry Pi's capabilities to the fullest, avert potential challenges, and extract its comprehensive computing dynamism.

Our methodology consists of:

- **Code Repository Acquisition:** We obtain code repositories from GitHub to understand their operation and performance, leveraging the extensive resources of open-source projects.
- **Performance Analysis:** We employ the Linux tool, perf[1] [15], to analyze performance metrics. Key metrics include context switches, cache usage, branch predictions, and more (detailed metrics can be found in Appendix A).
- **Parallelization Exploration:** We explore OpenMP and multithreading to enhance processing times. OpenMP offers a high-level approach, while multithreading provides more control and flexibility (detailed discussions on OpenMP and multithreading are available in Appendix B).
- **Optimizations:** Our aim is to reduce execution times, optimize memory access and utilize the cache to the maximum. We focus on direct code optimizations and strategies to optimize memory usage, utilizing tools like perf to guide our strategy(We cover compiler optimizations also in Appendix B).

To encapsulate, our investigative framework meticulously dissects the performance nuances of our cryptographic implementations. Moving past superficial execution metrics, we aim for a profound understanding, driving impactful performance uplifts.

3.2 AES

AES, in its different modes, such as ECB, CBC, and CFB, serves as an exemplary baseline to gauge the performance of Raspberry Pi devices due to its cryptographic importance and broad applicability across various sectors, including data security, digital communication, and e-commerce. A very known example is TLS.

Several reasons underscore the value of AES for performance evaluation on Raspberry Pi:

- **Cryptographic Complexity:** AES involves intricate cryptographic operations and data transformations. By gauging the performance of AES, one can fathom Raspberry Pi's capability to manage sophisticated cryptographic tasks and pinpoint potential enhancement zones.

[1] https://github.com/torvalds/linux/tree/master/tools/perf

- **Memory Access Patterns:** Implementing AES, particularly in modes like CBC and CFB, necessitates the continuous access and modification of data blocks. This exerts notable demands on the memory systems, including RAM and cache levels. Evaluating memory and cache behaviors during AES operations can shed light on Raspberry Pi's efficiency concerning memory accesses and offer hints for further optimizations.

- **Inherent Serial Dependency:** Modes like CBC introduce serial dependencies where the outcome of one encryption stage influences the next. This serial nature offers insights into Raspberry Pi's performance in scenarios where parallelism is not immediately possible, thereby assessing its single-core capabilities.

- **Pervasiveness in Real-World Applications:** AES is not just a theoretical concept; it is a cornerstone in modern encryption practices. Its application spans from securing online transactions to encrypting sensitive data. By evaluating AES performance on Raspberry Pi, we glimpse how these devices might fare in real-world data protection scenarios that lean heavily on encryption.

- **Parallelism Exploration in ECB:** Unlike CBC, the ECB mode offers a degree of parallelism since each block encryption is independent. This allows us to understand the multi-core potential of Raspberry Pi devices in cryptographic operations.

In summation, choosing AES in its various modes provides a multi-faceted perspective on Raspberry Pi's performance spectrum. Not only does this examination offer insights into cryptographic tasks, but it also lays a foundation for understanding and optimizing other data-intensive operations on Raspberry Pi.

3.3 Miller-Rabin

The Miller-Rabin primality test [12] offers a robust baseline for assessing the performance of Raspberry Pi devices due to its algorithmic sophistication and central role in numerous areas, notably in cryptography and number theory.

Here is why the Miller-Rabin test is a suitable choice for performance scrutiny on Raspberry Pi:

- **Algorithmic Depth:** The Miller-Rabin test is not a mere arithmetic operation. It is a probabilistic algorithm that involves modular exponentiation and squaring

sequences. Evaluating Raspberry Pi on this test helps discern its competence in handling complex algorithmic processes, highlighting any performance impediments or optimization zones.

- **Memory and Computational Intensity:** As numbers being tested increase in size, the computational intensity of the Miller-Rabin test rises, demanding more from the memory and computational subsystems. By examining the device's behavior during these tests, we can infer memory usage efficiency, cache behaviors, and potential bottlenecks specific to large-scale arithmetic operations.

- **Serial Nature with Iterative Checks:** The probabilistic nature of the Miller-Rabin test means it is often iterated multiple times to reduce the chances of a false positive. This repetitive process offers a window into the Raspberry Pi's stamina and consistency over prolonged computational tasks.

- **Real-World Relevance:** Primality testing is not an academic whim. It is fundamental in cryptography, especially in RSA encryption, where large prime numbers are vital. Analyzing Miller-Rabin's performance on Raspberry Pi not only gauges a specific algorithm but hints at the device's suitability for cryptographic operations and security applications.

- **Inherent Scalability Challenges:** With Miller-Rabin, the challenge is not just about large numbers and the inherent difficulty in verifying the primality of numbers as they grow. This allows for a nuanced understanding of how Raspberry Pi scales concerning algorithmic complexity and not just data volume.

In essence, leveraging the Miller-Rabin primality test as a performance benchmark provides a multifaceted examination of Raspberry Pi's capabilities. This analysis not only deepens our understanding of the device's algorithmic handling prowess but also offers a roadmap for enhancing performance across tasks of similar computational heft.

Chapter 4

Implementation and Evaluation

4.1 ECB Optimizations

ECB (Electronic Code Book) mode of operation encrypts each plaintext block independently to produce its corresponding ciphertext block. As a result, identical plaintext blocks yield identical ciphertext blocks, which can reveal patterns and make them less secure than other modes. In the analysis of ECB in subsections 4.1.1 and 4.1.2, we will focus on a sample of the data: a 115 MB file with a 256-bit key length. You can find more details for various key and file sizes in Tables C.1 through C.16. They all show the same patterns we will talk about here.

4.1.1 Software and Compiler Optimizations

This subsection delves into the ECB mode's initial performance without enhancements, followed by the integration of OpenMP, multithreading by parallelizing the blocks independently, and following with the O2 and O3 optimizations layered on top (all of these optimizations are explained in detail in Appendix B). This foundational understanding sets the stage for comparisons against enhanced versions in later sections. The initial, unenhanced ECB mode sets our benchmark, informing the forthcoming enhancement techniques. Moreover, this part offers detailed insights into each metric represented in the tables, laying the groundwork for comprehending the performance indicators. It is worth noting that we will only explain the metrics in detail in this subsection and assume a prior understanding of these metrics. This structure lets us focus squarely on the enhancements and their significance without rehashing metric definitions. In essence, this section primes the reader with the metrics' knowledge,

Table 4.1: General Benchmarks for ECB - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	context-switches	28,702	19,124	17,849	6,717	3,739	3,526	3,887	2,604	2,517
	page-faults	121,178	121,192	128,935	121,178	121,192	256,047	121,180	121,193	266,502
	cycles	602,541,850,714	634,059,935,557	621,873,389,204	142,682,634,897	142,159,101,868	140,957,534,552	81,615,068,371	81,406,782,029	81,002,619,423
	instructions	1,123,087,941,695	1,119,958,797,119	1,136,960,763,811	280,130,906,013	278,955,889,062	277,091,457,148	148,217,916,301	147,808,250,623	142,817,298,715
	time elapsed (s)	337.322497838	155.143214603	138.480971156	79.900445352	32.988503751	33.064546579	45.719792177	24.358153349	24.721736912
	cache-misses	52,289,636	60,661,726	73,691,828	32,688,667	28,850,760	50,348,042	27,654,673	25,378,853	118,501,650
	mem_access	472,304,431,263	478,799,154,548	473,733,041,368	102,685,385,712	101,288,498,258	102,560,951,180	72,922,341,565	71,739,177,363	71,383,071,735
	mem_access_rd	376,885,312,204	371,965,018,395	365,448,924,637	70,302,310,478	70,438,112,976	69,474,033,773	49,295,653,349	49,105,111,163	49,359,391,158
	mem_access_wr	95,362,031,528	95,110,390,850	92,501,373,711	31,782,047,347	31,840,813,647	31,556,731,660	22,913,136,630	22,878,557,275	23,213,878,033

fostering a more precise grasp of the upcoming discussions on enhancement approaches. Table 4.1 offers insights into the impact of various optimizations. Starting with context switches, a context switch occurs when the operating system's scheduler changes the process running on a CPU. This is necessary because there are typically many more processes that want to run than there are CPUs to run them. So the OS uses a scheduling algorithm to determine which process should run at any given time. In our data, compared to normal execution, OpenMP optimization reduced context switches by approximately 33% (from 28,702 to 19,124). With multithreading, there was a further decrease to 17,849. This suggests that both optimizations, especially multithreading, can decrease CPU task switching significantly. Page faults happen when a program attempts to access a page of memory not in its current set, causing the operating system to fetch it from the disk. An increase in page faults from the normal to the multithreaded version is observed in all the cases. This can be attributed to each thread in multithreading requiring its own stack and possible inefficiencies in CPU prefetching. Cache memory is a small amount of fast memory located on or very close to the CPU that stores frequently accessed data, helping speed up retrieval times. A cache miss occurs when the CPU checks for data in the cache, and the data is not there. Consequently, the CPU has to fetch the data from the main memory, which is significantly slower than retrieving it from the cache. This results in a delay in processing, leading to reduced performance. OpenMP and multithreading in the normal mode increase cache misses by 15% and 41%, respectively. But under O3 optimization, multithreaded execution sees cache misses spike to more than 4 times that of the normal version. This suggests potential cache thrashing due to concurrent threads in multithreading. For the elapsed

Table 4.2: L1 Cache Benchmarks for ECB - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	472,571,592,047	478,507,053,355	473,626,164,448	102,786,651,840	101,715,684,412	102,110,946,250	73,061,964,891	72,378,892,097	71,693,795,249
	l1d_cache_refill	52,434,218	60,611,149	70,382,295	31,749,761	28,341,640	48,825,798	26,801,938	24,902,010	117,352,356
	l1d_cache_rd	376,509,878,087	375,971,335,221	357,566,019,196	70,353,988,667	70,336,815,156	69,141,351,512	49,304,626,807	49,122,489,337	49,005,410,378
	l1d_cache_wr	95,448,338,486	96,178,550,129	89,971,630,866	31,986,899,253	31,817,869,815	31,587,341,269	23,026,123,136	22,894,420,470	23,105,076,297
	l1d_cache_wb	14,096,063	16,284,619	27,300,144	12,490,045	12,028,256	27,737,771	11,920,590	12,014,885	46,186,858
	l1d_tlb_refill	19,179,740	25,103,077	22,647,425	10,829,965	10,604,238	19,222,191	9,797,759	10,175,511	18,828,926
	l1d_cache_inval	50,051	524,078	5,779,199	15,994	144,719	11,952,304	6,984	118,571	12,214,268
	L1-icache-loads	359,231,099,763	366,132,417,144	343,818,168,724	94,404,311,636	93,031,128,685	93,302,014,346	46,473,470,474	45,827,867,004	45,394,059,040
	l1i_cache_refill	73,330,554	72,583,912	61,547,292	26,757,576	22,154,366	28,805,352	20,213,146	18,540,890	25,562,914
	l1i_tlb_refill	2,260,430	1,953,270	1,864,696	519,111	514,553	678,726	521,371	363,551	598,461

time, compiler optimizations drastically reduce execution time. For instance, the O3 version completes in approximately 13.5% of the time of the normal version.

In Table 4.2, we examine L1 Cache benchmarks. The L1 cache, comprising the L1 data cache (L1d) and the L1 instruction cache (L1i), is the initial level in the memory hierarchy. Events we are evaluating include L1-dcache-loads: Load instructions accessing the L1d cache. l1d_cache_refill: Refills of the L1d cache from a higher cache level due to misses. l1d_cache_rd: Read operations serviced by L1d. l1d_cache_wr: Write operations to L1d. l1d_cache_wb: Process where modified cache data writes back to main memory. l1d_tlb_refill: Refills of the Translation Lookaside Buffer (TLB) that aids in address translation. l1d_cache_inval: Invalidations of L1d cache data due to alterations in main memory. L1-icache-loads: Fetches from L1i cache. l1i_cache_refill: L1i cache refills from higher levels. And finally, l1i_tlb_refill: TLB refills for L1i. For L1 data cache loads, there is a 1.26% augmentation from the standard to OpenMP and a subsequent decrease of 0.99% with multithreading. The compiler optimizations O2 and O3 induce substantial reductions of approximately 78.22% and 84.56% from the normal state, respectively. This suggests heightened cache utility with these optimizations. The number of L1 data cache TLB refills experiences a 30.87% spike with OpenMP over the normal and a mild reduction to 18.09% with multithreading. A higher number of TLB refills often points to more cache misses. It means the system frequently tries to access memory locations not currently in the cache, prompting the need to fetch them from the main memory or another cache level. This could impact performance, as cache accesses are faster than other types of memory access. For L1 instruction cache metrics, compiler optimizations usher in pronounced declines, underscoring enhanced

Table 4.3: L2 Cache Benchmarks for ECB - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	229,955,956	276,963,942	327,751,901	147,621,731	188,535,662	226,647,758	130,002,316	129,650,346	376,885,467
	l2d_cache_refill	18,620,613	26,261,436	33,648,372	20,688,591	40,364,713	21,387,296	20,418,709	21,300,628	33,229,045
	l2d_cache_rd	179,989,353	213,663,654	198,939,946	109,369,222	92,161,551	159,153,582	117,753,608	125,386,404	198,960,323
	l2d_cache_wr	74,369,782	97,141,629	95,340,041	62,112,689	48,046,758	98,108,509	60,692,477	75,611,685	145,027,143
	l2d_cache_wb	5,271,614	7,669,235	10,245,482	6,120,566	6,758,029	6,241,214	6,519,856	6,421,028	8,011,591
	l2d_cache_inval	0	0	0	0	0	0	0	0	0

operational efficacy. Multithreading entails greater L1d TLB refills than its standard counterpart. This could be attributed to each thread's individual L1 cache. As they work on diverse data, the CPU has to refresh the TLB more often when transitioning between threads. Contrastingly, fewer L1i TLB refills in multithreaded operations suggest threads often share the same set of instructions. This communal utilization allows for shared memory addresses in the TLB, leading to fewer refills. This rendition focuses on the differences and their ramifications and further elucidates the implications of higher TLB refills.

Referring to Table 4.3 focusing on L2 Cache benchmarks: The L2 cache, larger but slower than the L1, acts as a secondary buffering stage. While the L1 cache is bifurcated into data and instruction components (L1d and L1i), the L2 cache is unified. Regarding L2 data cache utilization, the transition from normal to OpenMP sees an increase of about 20.4%. There is an 18.5% rise in moving to multithreading. With O3 optimization, however, multithreading leads to a significant 189.9% increase compared to the O3 normal. For L2 data cache refills, OpenMP from normal increases by 40.9% and multithreading by 28.8%. O2 with OpenMP uniquely surges by 95.1% compared to O2 normal, while other scenarios hover around a consistent 20 million refills. L2 cache reads indicate the number of read operations (or events) where the CPU tried to retrieve data from the L2 data cache. A growth of 18.7% is observed from normal to OpenMP, with a decrease of 6.9% in multithreading. O2 and O3 optimizations drop the number, with multithreading under O2 observing a rise of 45.5%. The higher cache used when using multithreading with O3 might be because each thread has its own L1 cache. If different threads are working on different data, the CPU might need to update the cache more often when it switches between threads. Also, the increase in L2 cache writes in this setup could be because the threads are talking to each other more or syncing up.

Branch prediction is a technique modern processors use to enhance speed, at-

Table 4.4: Branch Predictions and Speculation Benchmarks for ECB - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	65,054,116,184	65,457,627,240	64,231,942,793	48,831,557,909	48,242,550,068	48,412,182,212	16,226,796,596	16,151,436,058	16,224,674,379
	branch-misses	296,532,692	346,674,307	233,240,049	333,937,246	265,400,423	253,962,835	137,094,726	134,136,330	133,186,142
	br_immed_spec	57,954,968,932	58,643,735,463	57,257,007,069	47,154,425,825	46,521,743,435	47,224,055,162	14,400,672,088	14,179,758,287	14,103,659,296
	br_indirect_spec	7,182,103,929	7,308,312,386	7,176,326,583	1,772,669,791	1,745,784,013	1,835,821,822	1,983,831,824	1,966,730,304	1,973,691,337
	br_return_spec	5,622,934,087	5,717,895,781	5,617,709,700	1,139,034,627	1,124,795,281	1,194,218,826	1,348,334,788	1,334,491,759	1,353,260,759

tempting to foresee the outcome of operations before they are known. This efficient prediction aids in smooth operation; however, inaccuracies can lead to performance setbacks. As per Table 4.4, the key benchmarks include br_pred, which represents the total branch predictions, branch-misses indicating the number of incorrect predictions, br_immed_spec detailing predictions with immediate outcomes, br_indirect_spec for predictions that need calculations, and br_return_spec for predictions associated with function call returns. Analyzing the O2 optimization level, the br_pred varied by roughly 1.23% between the highest (OpenMP) and lowest (Threads) counts. For branch misses, there was a 23.95% variation, with Threads recording the fewest and OpenMP the most. Differences in br_immed_spec, br_indirect_spec, and br_return_spec stood at 1.35%, 1.52%, and 4.85%, respectively. On the other hand, under the O3 optimization level, br_pred showcased a minor 0.47% variation across all optimizations. Differences for branch-misses, br_immed_spec, br_indirect_spec, and br_return_spec were found to be 2.92%, 2.09%, 1.01%, and 1.39% respectively. It is noteworthy that, regardless of the specific optimization applied, branch prediction numbers remain relatively consistent within each optimization level. This indicates that branch prediction, predominantly driven by hardware, may not be significantly influenced by these software or compiler optimizations. Moreover, the structure and complexity of the code in execution can impact the efficiency of branch prediction. Thus, while software enhancements can offer advantages, it is essential to simultaneously consider the inherent limitations of hardware when aiming for optimal execution efficiency.

4.1.2 Optimizations with Transparent Huge Pages

As mentioned in Sec. 2.3, THP increases the usage of the Cache and reduced page faults; hence the TLB covers larger memory addresses. This subsection employes the

Table 4.5: General Benchmarks for ECB with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

ECB 115MB

Benchmarks	**256 Bits Key**								
				O2			O3		
	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
context-switches	30,504	17,110	19,637	6,675	3,849	3,701	3,869	2,438	2,516
page-faults	5,187	4,915	4,843	5,668	5,650	5,353	4,691	4,819	6,945
cycles	609,085,929,053	636,805,919,593	598,591,402,318	140,496,290,559	140,733,751,392	139,641,138,671	79,810,414,994	80,167,531,477	78,680,033,145
instructions	1,123,877,453,163	1,124,889,785,800	1,111,290,166,538	278,587,693,591	277,551,952,811	273,487,410,888	146,204,186,781	145,182,018,752	138,214,426,726
time elapsed (s)	341.388934029	136.284585948	159.258333051	78.668983721	32.606163104	32.483307029	44.704459181	23.948011504	24.092003188
cache-misses	45,273,252	41,013,637	67,136,702	24,349,980	25,669,214	48,582,441	26,852,606	34,795,024	39,626,246
mem_access	472,477,598,196	469,078,101,976	456,326,677,253	101,869,321,940	100,987,409,481	102,017,381,830	72,478,634,726	70,991,926,541	71,092,625,572
mem_access_rd	376,826,832,543	379,984,701,892	371,381,834,932	70,457,091,512	70,218,036,576	68,731,600,968	49,540,380,418	48,899,273,837	48,808,140,173
mem_access_wr	95,514,064,038	96,673,862,713	91,985,851,879	31,675,318,054	31,569,065,592	30,855,533,811	22,741,307,043	22,385,822,045	22,219,083,889

optimization in Sec 4.1.1 with THP and compare it. Table 4.5 offers insight into general benchmarks for ECB with Transparent Huge Pages. There is a significant reduction in time elapsed across all scenarios when shifting from the Normal version to the optimized 'O2' and 'O3' versions. For example, in the Normal version, the time elapsed shows a reduction of about 77% from Normal to O2 and an additional 43% decrease from O2 to O3. This underlines the effectiveness of software and compiler optimizations. Cache misses notably increase in the Threads version, especially compared to the Normal and OpenMP versions. In the transition from the Normal to Threads version, cache misses increase by around 48% for the O3 optimization level. This can be attributed to the inherent nature of multithreaded programs: multiple threads accessing data simultaneously can lead to cache invalidations, hence more cache misses. Nevertheless, the concurrent execution of threads compensates for this, as evidenced by the reduction in total execution time. Regarding page faults, using Transparent Huge Pages (THP) clearly offers an advantage. In the O3 optimized version for the Normal scenario, there is a 10% decrease in page faults when using THP. Comparing with results from Table 4.1, there is a pronounced reduction of around 96% in page faults for the O3 optimized Normal mode with THP. This emphasizes THP's ability to minimize the total number of pages, decreasing the chance of a page fault. Memory accesses, reading, and writing also show a marked decline from the Normal to the O2 and O3 versions, affirming the potency of these optimizations. While multithreaded versions have higher cache misses, performance enhancements from concurrent thread execution outweigh this shortcoming.

In analyzing L1 cache operations with Transparent Huge Pages (THP) in Table 4.6,

Table 4.6: L1 Cache Benchmarks for ECB with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	472,448,292,546	472,125,093,505	460,769,252,688	24,552,975	26,117,381	47,338,705	26,913,621	33,327,950	39,381,021
	l1d_cache_refill	45,205,119	40,375,044	69,132,981	70,235,876,070	69,933,265,571	68,491,926,984	49,171,100,340	48,707,085,795	47,946,389,444
	l1d_cache_rd	376,685,069,057	380,223,686,364	364,929,871,912	31,755,927,876	31,374,783,234	30,947,314,204	22,765,444,714	22,330,621,924	22,150,851,264
	l1d_cache_wr	95,312,078,933	96,587,814,139	89,984,829,839	11,134,803	11,813,882	22,572,160	15,342,054	13,853,517	22,270,894
	l1d_cache_wb	11,741,787	10,021,313	20,062,070	5,695,893	5,598,782	5,241,558	4,626,865	4,538,217	4,730,572
	l1d_tlb_refill	15,364,073	15,749,884	17,360,964	5,695,893	5,598,782	5,241,558	4,626,865	4,538,217	4,730,572
	l1d_cache_inval	59,006	360,407	1,264,641	8,481	72,901	4,546,171	9,018	22,428	4,899,114
	L1-icache-loads	359,632,289,614	360,057,857,483	334,572,578,100	92,419,709,867	92,669,313,830	92,025,640,233	45,219,560,454	44,438,721,369	43,854,593,142
	l1i_cache_refill	48,985,535	38,683,721	44,127,146	11,874,333	10,264,955	8,994,152	8,527,357	6,869,184	6,955,588
	l1i_tlb_refill	1,270,526	2,702,383	1,529,233	351,686	370,921	325,604	303,561	330,086	284,246

Table 4.7: L2 Cache Benchmarks for ECB with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	183,520,338	177,530,923	264,840,315	106,560,663	114,833,474	191,410,107	123,166,580	151,041,924	152,314,643
	l2d_cache_refill	19,552,420	23,100,034	32,686,335	24,308,405	24,070,917	30,389,660	23,849,214	29,558,828	26,916,875
	l2d_cache_rd	138,622,921	126,302,686	232,119,183	61,124,542	77,157,573	97,164,276	58,854,319	69,697,692	87,730,650
	l2d_cache_wr	58,794,511	51,633,640	125,966,731	36,511,938	44,098,983	65,115,236	37,298,280	62,904,305	65,581,190
	l2d_cache_wb	6,499,577	7,597,516	10,620,081	7,033,314	8,460,739	9,912,486	7,639,295	9,201,985	11,174,711
	l2d_cache_inval	0	0	0	0	0	0	0	0	0

multithreading across all optimizations showed a consistent increase in L1-dcache-loads and L1-icache-loads. This suggests a broader cache footprint due to diverse thread data sets, with a near 50% reduction in L1-dcache-loads for O3 threads when compared to previous results from Table 4.2 (71,693,795,249 to 39,381,021). Cache invalidations, however, surged under THP, attributable to false sharing risks in larger page sizes. Specifically, invalidations in O3 threads went from the lowest at 6,984 to 9,018 with THP. In terms of TLB refills, multithreading resulted in more l1d TLB refills but less for l1i, a possible outcome of thread commonality in instruction execution. Compared to prior data, a reduction of about 50% in TLB refills marks THP's efficiency.

Turning to L2 cache operations, performance under THP in Table 4.7 showcased significant improvements. For instance, the l2d_cache metric in O3 optimization with threads under THP was recorded at 152,314,643, starkly contrasting to the 376,885,467 observed in standard pages (Table 4.3). The l2d_cache_rd and l2d_cache_wr presented optimal figures with THP in O2 optimization using OpenMP, suggesting THP's effec-

Table 4.8: Branch Predictions and Speculation Benchmarks for ECB with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**256 Bits Key**								
					O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	84,666,814,422	85,293,075,241	85,412,236,062	48,345,787,488	48,186,371,101	47,642,529,687	15,781,358,151	15,731,396,912	15,096,099,851
	branch-misses	429,254,884	430,397,280	363,230,494	244,542,276	250,822,820	261,315,538	133,178,712	130,595,116	127,539,054
	br_immed_spec	76,725,988,906	77,030,579,954	75,136,950,440	46,508,322,851	46,196,410,542	46,269,947,985	13,940,357,202	13,754,426,613	13,653,026,782
	br_indirect_spec	8,121,412,976	8,227,583,997	8,065,470,140	1,695,514,925	1,643,602,798	1,690,425,697	1,901,349,055	1,860,917,134	1,862,097,383
	br_return_spec	6,300,863,919	6,381,566,687	6,280,496,183	1,088,532,754	1,051,184,929	1,089,952,877	1,292,905,804	1,262,413,117	1,273,146,656

tiveness in enhancing cache read/write operations. There were no noticeable differences in other categories, implying that THP's impact may be limited to specific cache operations.

In comparing the benchmarks from Table 4.4 and Table 4.8, the former generally outperforms the latter across all categories: Normal, OpenMP, and Threads. Notably, in the Normal mode, branch misses have increased by approximately 45%, rising from 296,532,692 to 429,254,884. Such an increase is also observed in the OpenMP and Threads modes, pointing to more branch misses. These misses offer insights into the accuracy of the branch predictor, with a higher number indicating lower prediction accuracy. The number of immediate branch speculations, known as br_immed_spec, has also decreased in Table 4.4 relative to Table 4.8. This type of speculation involves control flow changes where the jump target is known immediately, and a decrease indicates better branch prediction rates. The same trend is observed for indirect branch speculations (br_indirect_spec), which concern jumps where the target is known indirectly, such as through a register. Furthermore, there is a noted reduction in the speculative branches following a return instruction, or br_return_spec, in Table 4.4, suggesting enhanced performance in the standard page benchmarks. The reasons for the superior performance in standard page benchmarks could be varied, perhaps due to different memory usage or thread management in OpenMP and Threads. It is worth noting that ECB inherently offers high parallelism, which is beneficial for systems equipped with multi-core processors or vector instruction sets. As for the influence of Transparent Huge Pages (THP), it can enhance performance by reducing the overhead of memory management through the use of larger pages, leading to fewer TLB misses.

Table 4.9: General Benchmarks for CBC - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	context-switches	42,922	34,667	14,685	12,347	6,478	5,539
	page-faults	242,234	242,249	242,234	242,249	242,234	242,250
	cycles	902,108,769,086	919,755,772,047	223,438,018,980	222,105,527,392	133,797,182,398	133,885,379,555
	instructions	1,652,654,882,325	1,657,579,070,441	436,403,509,738	435,323,267,979	242,865,454,325	237,014,031,386
	seconds	505.010304178	361.558635826	126.170184770	94.537819957	74.985033373	62.201525668
	cache-misses	104,704,837	106,684,448	64,280,399	86,808,285	48,547,839	50,183,964
	mem_access	704,040,818,057	700,960,551,103	161,068,206,366	160,017,003,390	117,500,022,693	116,279,018,259
	mem_access_rd	555,973,955,674	556,728,252,604	110,000,812,809	109,794,562,690	79,410,913,107	78,577,298,405
	mem_access_wr	148,311,857,594	148,335,461,233	51,842,343,924	51,735,079,781	38,331,994,936	38,126,350,967

4.2 CBC Optimizations

As we explained in Sec 2.2, in CBC (Cipher Block Chaining) mode, each plaintext block is XORed with the previous ciphertext block before being encrypted, creating a chain of dependency among blocks. In contrast, ECB (Electronic Code Book) mode encrypts each plaintext block independently. We will focus mainly on the 128-bit key length with a 676 MB file while ensuring that the observed trends remain consistent across varying file sizes. Tables D.1 to D.16 provides extensive insights into all the different key and file sizes and presents the same trends that will be discussed in this section.

4.2.1 Software and Compiler Optimizations

We turn our attention to the Cipher Block Chaining (CBC) mode of operation without any added performance enhancements. At its core, CBC takes plaintext blocks and encrypts them sequentially, using the previous block's ciphertext to influence the encryption of the current block. CBC inherently poses challenges for parallelization because each block's encryption depends on the result of the previous one. This sequential dependency can slow the processing speed, especially when handling large volumes of data. Setting out with this raw version of the CBC mode serves a two-fold purpose. First, it allows us to grasp the foundational workings of the method without any alterations. Second, it provides a performance baseline, helping to benchmark and quantify the gains achieved through subsequent optimizations. As we progress through this study, we will layer various optimization strategies onto the CBC algorithm and

assess their impact. These will encompass techniques like pipelining to mitigate the inherent sequential nature of CBC, as well as compiler-driven improvements such as O2 and O3 optimizations. The aim is to compare this foundational CBC implementation against the augmented versions, drawing clear contrasts and gauging the merit of each optimization in enhancing the CBC mode's efficiency. To begin with, Table 4.9 highlights several relevant benchmarks that enhance our understanding of the effectiveness of these optimizations. First, a noticeable improvement is seen in the context switches. There is a substantial drop from 42,922 under normal circumstances to just 5,539 with the O3 level optimization and OpenMP. This drastic reduction indicates less processor time spent on task-switching, streamlining the computational resources to the tasks. Regarding page faults, there is no evident change between the different settings, with the numbers remaining consistent. However, the importance of this parameter lies in memory management, and maintaining a steady, low number of page faults is a positive indicator. The execution time is markedly improved from the original 505.01 seconds to 62.20 seconds under OpenMP and O3 optimizations. This significant decrease in execution time exhibits the optimizations' potency in enhancing the CBC mode's efficiency in AES. Notably, cache misses are minimized under the O3 level optimization with 48,547,839 misses, considerably lower than the rest. This reflects a more effective utilization of cache memory, promoting quicker memory access and better performance. Memory accesses, both read (mem_access_rd) and write (mem_access_wr), are also minimized in the O3 level optimization. This demonstrates more efficient memory management, resulting in a lower need for memory bandwidth and potentially reducing the system's latency. The beneficial effect of these optimizations is apparent, leading to a more efficient system.

Moving on into Table 4.10 representing L1 cache-related benchmarks for CBC. A key observation from the table is the decreasing trend in L1-dcache-loads, indicating the number of attempts made by the processor to read data from the L1 data cache. When viewed in the context of the previous tables' cache misses, this decrease suggests an improvement in data availability in the cache, reducing the necessity to load data. A similar trend can be observed in the l1d_cache_refill category, signifying the refill operations that occur when requested data is not found in the cache. The decrement in the number of refills with O3 optimization further illustrates the enhancements to cache efficiency, as fewer refills infer a lower frequency of cache misses. As for l1d_tlb_refill, it portrays the refill operations for the Translation Lookaside Buffer (TLB), a memory cache that holds recent translations of virtual memory to physical addresses. The

Table 4.10: L1 Cache Benchmarks for CBC - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	L1-dcache-loads	703,949,676,276	699,135,671,337	162,044,278,113	161,221,435,014	117,589,426,754	115,871,395,278
	l1d_cache_refill	107,654,471	108,940,600	61,215,652	85,834,488	47,778,118	53,208,308
	l1d_cache_rd	555,663,499,938	550,336,745,015	109,476,765,536	109,167,633,629	79,246,726,657	79,272,166,772
	l1d_cache_wr	148,299,780,923	147,108,286,255	51,825,508,787	51,585,129,644	38,293,154,418	37,892,054,783
	l1d_cache_wb	33,298,472	34,080,369	26,421,733	40,304,546	23,469,898	32,495,122
	l1d_tlb_refill	40,214,915	73,322,894	25,600,473	26,747,663	22,153,285	24,521,461
	l1d_cache_inval	88,105	416,226	15,343	139,680	10,491	111,956
	L1-icache-loads	538,573,692,016	514,838,481,384	148,044,931,741	145,633,047,284	78,475,532,381	76,903,792,875
	l1i_cache_refill	126,953,024	123,635,343	59,146,105	54,863,559	38,597,703	39,727,168
	l1i_tlb_refill	7,062,313	3,330,298	1,607,361	1,432,010	579,313	1,246,113

Table 4.11: L2 Cache Benchmarks for CBC - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	l2d_cache	504,325,816	556,978,454	354,065,786	421,044,500	301,564,015	282,047,158
	l2d_cache_refill	53,166,822	51,176,328	61,688,751	70,780,594	60,678,242	53,497,486
	l2d_cache_rd	324,101,881	423,589,888	202,627,623	219,272,209	157,386,141	192,473,556
	l2d_cache_wr	146,276,748	144,485,838	98,809,710	123,105,904	88,789,738	104,514,503
	l2d_cache_wb	11,577,767	16,614,301	14,010,035	18,498,052	13,737,747	19,987,727
	l2d_cache_inval	0	0	0	0	0	0

drop in TLB refills denotes an improvement in address translation efficiency. A lower refill count corresponds to fewer TLB misses; hence fewer CPU cycles are wasted on address translations. The data on l1d_cache_inval, or cache invalidations, also reveals an interesting trend. Cache invalidations occur when the cache needs to discard certain data due to changes in the main memory. The table shows that these invalidations are less frequent under O3 optimization, indicating that the system experiences fewer cache coherence issues. These observations show that software and compiler optimizations, particularly the O3 level, can significantly improve cache and memory management. These enhancements translate into reduced computational waste and improved execution efficiency for CBC in AES.

Moreover, Table 4.11 represents L2 cache-related benchmarks for CBC. The l2d_cache event shows a noticeable decline. This downward trend indicates improved memory management, where data is more effectively stored and retrieved from the faster L1 cache, thus reducing the reliance on the slower L2 cache. The efficient utilization

Table 4.12: Branch Predictions and Speculation Benchmarks for CBC - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	br_pred	131,569,364,108	130,931,061,838	76,850,503,736	76,724,135,696	29,801,321,067	29,434,922,230
	branch-misses	733,600,566	626,959,443	548,677,622	407,681,124	272,279,207	265,333,022
	br_immed_spec	117,381,993,979	116,244,743,200	73,034,999,777	72,668,396,892	26,418,686,928	26,042,053,211
	br_indirect_spec	14,389,872,568	14,366,560,276	3,416,966,337	3,476,139,916	3,718,955,622	3,666,854,239
	br_return_spec	11,286,829,476	11,269,211,214	2,163,470,390	2,229,191,756	2,464,413,999	2,423,406,656

of the L1 cache can be credited to the advanced compiler optimizations, particularly O3. The slightly increased the l2d_cache_refills', which signifies instances when the L2 cache needs to fetch data from the main memory due to cache misses, seems counterintuitive given the reduction in L2 cache accesses. However, it can be explained by the higher incidence of L1 cache misses due to the shifting memory use patterns that come with compiler optimization. Despite the higher refill count, overall cache performance is still improved due to lower L2 cache access. Looking at l2d_cache_rd and l2d_cache_wr, we observe a decrease, indicating that less data is being read from or written into the L2 cache. This trend aligns with the improved L1 cache usage, minimizing the need for slower L2 cache interactions. Finally, l2d_cache_wb exhibits a small increase, indicating write-back operations where modified data is written back to the main memory from the cache. This might be due to the higher rate of data modifications under the O3 optimization level, requiring more frequent updates to maintain cache coherency.

Finally, Table 4.12 regarding Branch prediction and speculative executions. The br_pred event shows a decrease in the number of predictions made by the processors' branch predictor. A lower number suggests that the system's execution flow has become more straightforward or better optimized, thus requiring fewer branch predictions. Between the normal operation mode and O3 optimization, there is a significant reduction of about 77%, indicating a considerable leap in efficiency. The branch-misses field, representing instances where the processor's branch prediction was incorrect, also shows a decrease. This means that not only are fewer predictions being made, but those that are made are more accurate. Again comparing normal to O3, the number of branch misses has decreased by approximately 63%. The immediate, indirect, and return speculative branches are branches where the processor makes an educated guess about the flow of the program, allowing it to execute instructions ahead of time to optimize performance. The numbers decreasing for br_immed_spec indicates an improvement in the system's

Table 4.13: General Benchmarks for CBC with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	context-switches	43,413	35,353	14,291	8,460	6,383	5,722
	page-faults	32,216	5,705	8,430	6,367	11,296	5,707
	cycles	909,221,057,907	903,091,682,540	219,784,412,235	217,512,446,948	131,505,431,162	132,147,138,822
	instructions	1,647,996,120,743	1,597,401,618,099	431,854,910,281	430,533,358,095	240,481,464,759	236,343,313,699
	seconds	508.958926340	356.860918281	123.747516418	91.494814982	73.672802774	60.733652527
	cache-misses	87,253,338	76,430,360	62,709,712	72,587,217	46,836,593	42,664,023
	mem_access	706,864,124,820	693,695,012,052	160,912,671,364	159,031,541,815	116,378,084,293	114,260,838,455
	mem_access_rd	557,974,868,308	547,342,072,321	108,964,521,445	108,601,724,774	78,839,848,139	78,256,860,720
	mem_access_wr	148,690,921,331	142,929,930,537	51,047,956,310	50,788,728,550	37,962,039,820	37,605,796,743

ability to predict the program flow correctly, resulting in fewer speculative executions. Interestingly, for br_indirect_spec and br_return_spec, we see an increase when moving from the normal mode to O2 and then a slight decrease with O3. This may suggest that the O3 optimizations effectively managed some of the additional complexity introduced with O2 optimization.

4.2.2 Optimizations with Transparent Huge Pages

In comparing Tables 4.9 and 4.13, the use of THP in CBC operations results in notable optimizations. A significant drop in page faults is observed with THP, from 242,234 in the Normal mode of Table 4.9 to just 32,216 in Table 4.13, marking a decrease of around 86.7%. Page faults, which indicate time-intensive disk reads when a required page is not in memory, are substantially minimized with the efficient memory utilization of THP. Cache misses, events where data is not found in the cache, also show a decline. The reduction from the Normal mode to O3 in cache misses is about 46% for both operations in Table 4.13, signifying better cache efficiency, possibly due to the larger pages of THP. Other metrics, such as context switches, cycles, and instructions, manifest a consistent downward trend as we progress from Normal to O2 and O3 optimizations in both tables, underscoring the effectiveness of increased compiler optimization levels. For instance, cycles in the Normal mode were reduced by approximately 74.2% from Table 4.9 to Table 4.13 when using O3 optimization. While memory access metrics show variations, the differences are marginal, and they primarily serve as indicators of how often the system interacts with memory.

Comparing Table 4.10 with standard pages to Table 4.14 using Transparent Huge

Table 4.14: L1 Cache Benchmarks for CBC with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	L1-dcache-loads	706,092,109,906	689,395,272,892	160,491,628,370	158,919,485,523	116,111,149,305	115,741,414,680
	l1d_cache_refill	87,308,147	77,612,029	64,174,282	71,715,801	45,398,420	41,913,020
	l1d_cache_rd	558,096,683,021	543,753,114,397	109,153,655,807	107,622,480,306	78,359,036,715	77,646,594,855
	l1d_cache_wr	148,871,264,330	142,494,203,971	51,158,999,334	50,579,054,441	37,834,085,230	37,376,747,027
	l1d_cache_wb	29,199,481	27,658,277	30,313,404	37,060,106	23,781,096	27,923,198
	l1d_tlb_refill	26,179,424	34,546,385	11,414,425	9,525,043	10,273,300	8,414,093
	l1d_cache_inval	67,818	303,338	21,589	52,440	10,200	41,553
	L1-icache-loads	546,011,733,180	523,384,421,360	146,048,320,944	144,019,219,579	76,715,012,113	76,372,714,808
	l1i_cache_refill	77,629,295	67,339,887	31,992,437	17,308,062	16,462,130	13,891,769
	l1i_tlb_refill	2,725,296	2,023,355	944,308	662,175	440,093	683,257

Pages (THP) offers insights into L1 Cache Benchmarks for CBC. For the l1d_cache_refill event, there is a noticeable decrease across all configurations when using THP. In the O3 optimization with OpenMP, for instance, there is an approximate 21% decrease. This indicates fewer refills of the Level 1 data cache, which can lead to performance boosts. The l1d_tlb_refill event also showcases a decrease with THP. In the O3 optimized Normal mode, we witness about a 54% reduction, suggesting fewer TLB refills and improved performance. However, the l1d_cache_inval event displays inconsistent behavior. While some modes reveal a decrease, others see an uptick, hinting at the intricate nature of application behavior and memory management. For the remaining events, such as l1d_cache_rd, l1d_cache_wr, l1d_cache_wb, L1-icache-loads, l1i_cache_refill, and l1i_tlb_refill, the observed changes generally favor THP. Specifically, moving from Normal to O2 and O3 optimizations with THP usually leads to decreased values, pointing to enhanced cache efficiency. In summary, THP offers clear benefits in cache and memory management, reducing operations like cache and TLB refills and subsequently augmenting the performance of CBC operations when combined with software and compiler optimizations.

Upon comparison of Tables 4.11 and 4.15, several observations emerge regarding the impact of Transparent Huge Pages (THP) on L2 cache management for CBC operations. The **l2d_cache** event, which represents total accesses to the L2 data cache, significantly declined with THP. For instance, under O3 optimizations in OpenMP mode, there is an approximate 6% reduction, illustrating enhanced cache utilization. In the case of the l2d_cache_refill event, there was a reduction in most scenarios but a slight 9% increase under O3 in OpenMP mode, suggesting that aggressive O3 optimizations

Table 4.15: L2 Cache Benchmarks for CBC with Transparent Huge Pages- Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	l2d_cache	386,825,285	367,022,846	244,922,680	344,898,556	257,862,036	264,351,134
	l2d_cache_refill	51,171,710	57,559,786	39,928,374	68,103,624	58,895,803	65,329,862
	l2d_cache_rd	264,328,272	244,836,127	167,422,669	161,610,371	155,664,883	141,731,339
	l2d_cache_wr	132,691,461	115,244,042	126,589,341	109,171,912	90,543,291	91,201,517
	l2d_cache_wb	16,407,993	18,061,327	14,696,656	24,206,572	19,700,892	24,470,807
	l2d_cache_inval	0	0	0	0	0	0

Table 4.16: Branch Predictions and Speculation Benchmarks for CBC with Transparent Huge Pages - Software Optimizations and Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**					
				O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 676 MB	br_pred	131,711,861,128	131,239,577,256	75,984,852,945	75,611,936,592	29,225,128,788	28,911,666,799
	branch-misses	841,122,280	727,124,178	412,803,363	405,472,593	267,274,061	263,265,384
	br_immed_spec	117,698,261,917	117,238,853,207	72,936,982,290	72,075,189,301	25,513,037,563	24,988,981,021
	br_indirect_spec	14,316,561,464	14,383,199,554	3,363,246,064	3,261,513,422	3,545,046,317	3,460,607,296
	br_return_spec	11,215,758,627	11,231,467,141	2,118,565,135	2,055,598,911	2,334,611,539	2,278,988,645

might be prompting more cache evictions and subsequent refills. Read operations on the cache, l2d_cache_rd, diminished with the application of THP. Specifically, there is about a 26% reduction under O3 optimizations in OpenMP, indicating more efficient memory reads. Write operations, l2d_cache_wr, followed suit, with an observed 28% decline under O2 optimizations in normal mode. However, the l2d_cache_wb event, which measures write-back operations, showed a variable trend, likely influenced by different memory access patterns and strategies across optimization levels. It is worth noting that the l2d_cache_inval event, indicating invalidation operations, remained unchanged in both tables. To summarize, THP's introduction considerably refines cache management in CBC operations, marked by decreased cache accesses, refills, reads, and writes. Such efficient memory operations likely lead to minimized latencies, enhancing overall performance.

In comparing Tables 4.12 and 4.16, we can observe the effect of Transparent Huge Pages (THP) on the branch prediction and speculation processes in CBC operations. The br_pred event, representing predicted branches, shows a negligible reduction across all compiler optimizations, with no significant differences. The branch-misses event signifies a substantial decrease with THP, particularly a 14% decrease in OpenMP with

Table 4.17: General Benchmarks for Miller Rabin Method 1 - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
context-switches	58,495	75,428	24,592	81,602	26,168	95,716
page-faults	111	130	111	131	112	130
cycles	1,234,415,357,026	798,578,478,024	523,179,582,500	543,243,421,289	499,538,644,477	546,613,835,428
instructions	1,188,461,916,165	768,053,201,869	319,214,161,200	336,513,693,599	305,306,947,984	340,954,318,137
seconds	690.973606098	174.872790763	292.891665947	152.706866776	279.755920462	102.251458860
cache-misses	43,195,907	143,543,462	18,573,176	136,223,846	17,733,169	184,087,384
mem_access	704,421,235,236	449,798,394,925	13,500,726,299	20,012,514,519	12,886,807,200	21,643,334,377
mem_access_rd	485,799,074,899	308,509,758,524	8,096,262,286	11,191,841,533	7,741,230,029	11,998,997,077
mem_access_wr	218,613,758,931	141,636,172,485	5,405,649,555	8,747,167,979	5,127,601,561	9,640,021,754

O3 optimization, indicating enhanced branch prediction accuracy. The br_immed_spec event slightly decreases, while the br_indirect_spec fluctuates with minor increases in some cases. The br_return_spec event decreases overall, reflecting optimization in return predictions. Except for the mentioned changes, there are no other noticeable differences between the tables. In summary, THP's introduction mainly contributes to reducing branch misses and slight refinement in immediate speculation branches, enhancing the execution flow control and performance in CBC operations.

4.3 Miller Rabin

For the sake of generality, a representative subset of the data will be utilized in this analysis. In particular, method 2 will only utilize 5000 and $10,000$ numbers generated via the normal mode, multithreaded, and compiler optimizations. In method 2, $100,000$ and $1,000,000$ in the multithreaded version had problems with thread generation. We tried several thread pooling methods, but nothing worked. We spent almost a week figuring out how to find a 64-bit prime number using Miller-Rabin primality tests. The complete tables are provided in Appendix F and in Tables F.1 to F.8.

4.3.1 Software and Compiler Optimizations

To initiate our deep dive into primality tests, we started with the unoptimized Miller-Rabin test via Method 1. This inaugural approach to prime generation is characterized by manually constructing numbers with a predefined bit size, followed by an adjustment to ensure they're odd. Pivoting our focus to optimization, this base version serves as a crucial touchstone, painting a vivid picture of potential areas of enhancement. It offers

Table 4.18: General Benchmarks for Miller Rabin Method 2 - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
context-switches	444	916	12,532	25,011	12,587	360	10,858	22,011	207	324	10,218	22,345
page-faults	111	111	10,507	20,897	110	110	10,506	20,898	111	110	10,508	20,898
cycles	24,387,656,932	48,887,382,863	1,827,572,711	4,522,205,566	15,746,551,258	30,047,964,202	2,334,744,801	3,202,165,980	14,946,032,415	29,748,792,141	1,256,701,944	2,703,776,247
instructions	22,979,713,807	46,088,433,723	1,745,612,559	4,651,210,683	8,782,226,157	16,774,062,352	1,368,194,543	2,121,814,945	8,355,251,956	16,619,694,457	757,606,621	1,668,432,327
seconds	13.648183706	27.381384148	5.192182217	10.410437517	9.477200115	16.815506940	3.867477558	7.742098412	8.388323421	16.651904282	3.941819254	7.762215084
cache-misses	1,748,512	3,320,065	3,766,812	5,510,127	5,479,785	2,719,841	2,849,788	5,891,159	1,195,771	2,654,593	1,940,734	4,527,300
mem_access	13,753,289,642	27,572,563,770	1,342,009,809	2,782,432,076	304,311,694	209,647,146	150,366,029	282,432,077	100,750,290	211,525,429	106,463,880	229,255,753
mem_access_rd	9,512,144,400	19,070,146,411	713,792,838	1,649,074,326	179,133,460	107,953,558	50,825,428	95,563,892	53,172,324	108,365,969	42,232,462	116,402,434
mem_access_wr	4,247,573,810	8,515,632,490	348,445,702	823,238,695	154,846,123	100,770,887	50,203,552	93,328,921	48,972,073	102,987,010	40,122,711	116,734,170

a foundation from which we can ascertain the fruits of our optimization endeavors. As we advance, we will scrutinize various optimization techniques layered onto the Miller-Rabin test. Their efficacy will be assessed and compared against our unoptimized Method 1. Further, in the spirit of a comprehensive evaluation, Method 2 will be put under the microscope. We will weigh both methods against contemporary optimization standards such as OpenMP, multithreading, and the O2 and O3 levels, echoing the approach taken with the previous cryptographic methods. To initiate, Method 1, as presented in Table 4.17, showcases an increased number of context switches when multithreading is incorporated. This contrasts with Method 2 in Table 4.18, which, despite generating a considerably higher number of prime numbers, results in fewer context switches. When considering page faults, Method 1 remains consistent regardless of multithreading. Conversely, Method 2 witnesses a rise in multithreading. Shifting attention to software optimization, both methods, when optimized (O2 and O3), record reduced CPU cycles and instructions. In percentage terms, compared to the non-optimized versions, Method 1 experiences reductions of approximately 57% and 59% in CPU cycles for O2 and O3, respectively. Meanwhile, Method 2 exhibits larger variations. However, these optimizations also lead to an increase in cache misses in both methods, implying potential inefficiencies in memory access. Surprisingly, Method 2 reports notably fewer memory accesses than Method 1, indicating its efficiency in operation execution and its prowess in prime number generation. It is essential to highlight that despite multithreading's potential to speed up computations, it might introduce overhead through added context switches and cache misses. Simultaneously, software optimizations, while decreasing CPU cycles and instructions, may contribute to more cache misses. Notably, there are certain metrics, such as mem_access_wr, where no noticeable difference between the two methods in Tables 4.17 and 4.18 can be observed.

Table 4.19: L1 Cache Benchmarks for Miller Rabin Method 1 - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
L1-dcache-loads	704,400,891,841	450,878,012,091	13,504,936,737	20,013,939,627	12,862,396,501	21,561,262,925
l1d_cache_refill	42,812,866	143,944,398	18,612,310	135,529,715	17,878,597	183,761,238
l1d_cache_rd	485,734,174,547	308,887,536,822	8,109,070,126	11,222,194,538	7,739,684,977	11,986,546,163
l1d_cache_wr	218,608,505,382	141,760,544,954	5,407,588,502	8,744,449,694	5,128,397,243	9,603,428,602
l1d_cache_wb	5,491,311	55,984,070	3,108,641	52,108,987	2,228,658	71,930,326
l1d_tlb_refill	20,451,520	93,978,580	8,754,252	66,901,719	8,635,905	79,683,016
l1d_cache_inval	149,841	53,126,012	55,931	49,446,792	107,462	67,146,987
L1-icache-loads	922,621,685,314	588,741,694,691	335,867,680,412	345,812,403,837	315,639,255,920	328,433,283,868
l1i_cache_refill	77,010,001	90,136,018	32,652,999	75,961,509	33,476,564	87,081,448
l1i_tlb_refill	1,296,747	7,500,516	575,543	5,626,069	506,903	6,494,920

Table 4.20: L1 Cache Benchmarks for Miller Rabin Method 2 - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
L1-dcache-loads	13,756,514,268	27,582,430,500	1,935,685,747	2,921,968,108	323,429,729	205,220,800	120,958,512	286,091,403	107,962,048	204,179,245	153,263,079	278,134,474
l1d_cache_refill	1,753,967	3,257,598	3,616,081	5,298,176	5,607,449	2,657,981	2,595,063	6,668,958	1,208,947	2,625,170	3,131,879	5,623,390
l1d_cache_rd	9,505,225,209	19,059,815,626	921,059,058	1,917,821,206	184,735,993	108,546,001	66,155,575	90,531,944	54,294,437	110,240,270	42,347,310	96,762,130
l1d_cache_wr	4,247,863,133	8,513,658,858	315,079,159	882,470,851	151,063,698	100,032,166	61,280,155	86,901,007	51,396,628	100,005,074	54,384,393	80,621,857
l1d_cache_wb	450,283	660,622	1,962,360	3,016,743	1,393,381	683,675	1,408,728	3,736,626	336,677	798,242	1,635,307	3,103,266
l1d_tlb_refill	1,452,867	2,782,039	1,250,227	2,088,330	3,488,837	2,467,545	954,044	2,600,061	1,258,250	2,422,306	1,048,972	2,023,534
l1d_cache_inval	154,081	318,698	79,936	227,251	494,921	250,918	119,437	140,865	124,563	281,917	63,551	158,243
L1-icache-loads	18,309,854,831	36,707,741,861	2,105,881,962	3,232,108,795	9,754,608,045	19,427,420,195	956,840,662	2,038,035,988	9,324,095,932	18,541,403,374	649,925,608	1,344,219,295
l1i_cache_refill	4,567,346	9,194,113	3,680,792	5,417,605	14,595,601	6,889,128	2,540,033	5,378,060	3,456,003	6,875,394	1,990,097	4,256,907
l1i_tlb_refill	257,832	502,872	134,063	219,976	461,614	350,017	110,173	262,588	193,050	407,871	65,942	173,036

Looking at the L1 Cache data, there are apparent differences between the Miller Rabin Method 1 (4.19) and Method 2 (4.20). Method 1 starts with a very high 704.4 trillion cache loads. With multithreading, this number drops by about 36%. On the other hand, Method 2's numbers drop a lot more with multithreading, especially for the 10000 instances where it drops by 89%. This suggests that Method 2 is better at using resources. In Method 1, some numbers, like l1d_cache_refill, go up much with multithreading. However, Method 2 handles these better, with numbers decreasing by 63% for l1d_cache_refill and 41% for l1d_cache_inval for the 10000 instances. Other numbers, like l1d_cache_rd and l1d_cache_wr, show similar patterns. When we use more threads, Method 2 usually does better than Method 1. For the software changes, O2 and O3, the results are mixed. In Method 1, some numbers drop a lot, like a 98% drop for L1-dcache-loads. But other numbers go up. This tells us that these software changes can be good in some ways but not so good in others. Sometimes, using more threads can cause problems, maybe because of managing the threads. In short, both methods can be improved with more threads and software changes, but we need to look at each number closely and carefully.

Table 4.21: L2 Cache Benchmarks for Miller Rabin Method 1 - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
l2d_cache	172,512,036	407,499,082	73,811,616	368,212,774	73,144,080	475,963,615
l2d_cache_refill	503,884	1,862,594	242,751	1,747,561	362,547	1,890,600
l2d_cache_rd	128,678,141	273,783,286	55,161,425	243,451,764	53,296,134	309,632,644
l2d_cache_wr	43,849,053	135,962,757	18,887,043	122,254,374	18,527,311	165,111,998
l2d_cache_wb	62,413	256,928	32,691	237,169	49,731	262,564
l2d_cache_inval	0	0	0	0	0	0

Table 4.22: L2 Cache Benchmarks for Miller Rabin Method 2 - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
l2d_cache	8,768,808	17,446,566	15,026,289	25,586,722	28,304,853	13,795,036	11,697,717	25,232,543	6,519,618	13,931,882	8,995,104	22,430,695
l2d_cache_refill	101,081	349,200	2,511,540	4,901,677	2,915,789	221,871	2,381,305	4,898,604	167,674	220,285	1,723,980	4,181,885
l2d_cache_rd	7,135,609	13,947,293	5,306,276	16,516,000	23,260,117	11,212,579	7,243,050	13,378,162	5,136,253	11,018,946	7,602,049	13,087,624
l2d_cache_wr	1,865,289	3,414,106	2,071,764	5,444,251	5,667,010	2,873,953	2,892,592	3,951,314	1,278,408	2,785,252	2,839,939	5,756,707
l2d_cache_wb	13,751	48,095	929,418	2,185,969	316,448	33,947	1,157,676	1,864,215	17,903	28,237	663,437	1,676,650
l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Comparing L2 cache benchmarks between Miller Rabin Methods 1 (4.21) and 2 (4.22), significant disparities in cache behaviors are evident. In Method 1, the l2d_cache increases by 136% when multithreading is enabled. Yet, with O2 and O3 optimizations, there is a decline in cache loads by 57.2% and 57.6%, respectively. Contrastingly, in Method 2, the l2d_cache shows a surge under multithreading, especially noticeable at 71% for 5000 numbers. Optimizations O2 and O3 yield dramatic drops in cache loads, the most notable being a 76% drop for 10,000 numbers under O3. Cache refill behaviors differ between methods. In Method 1, multithreading causes an increase of 269%. Meanwhile, Method 2's multithreading boost is considerably steeper, peaking at 1300% for 10,000 numbers. Curiously, O2 and O3 optimizations in Method 2 amplify cache refills. Some benchmarks, such as l2d_cache_wr and l2d_cache_wb, fluctuate under multithreading and software optimizations across both tables. Yet, certain events like l2d_cache_inval consistently display no variation between both methods. In essence, while multithreading and software optimizations can enhance computational efficiency, they sometimes intensify system resource usage. This underpins the necessity for meticulous profiling and tactical deployment to optimize system performance.

When examining branch prediction and speculation benchmarks presented in Tables 4.23 and 4.24, several patterns emerge. For the br_pred benchmark in Method 1 (Table 4.23), the application of multithreading resulted in a roughly 33% reduction from its standard state. Optimizations O2 and O3 showed smaller declines, around 33% and 36%, respectively. In stark contrast, Method 2 (Table 4.24) demonstrates that,

Table 4.23: Branch Prediction and Speculation Benchmarks for Miller Rabin Method 1 - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
br_pred	119,092,387,162	79,971,964,510	80,135,841,537	84,038,400,531	76,636,388,703	85,110,806,528
branch-misses	22,655,503,617	14,264,643,956	17,955,338,681	18,048,707,009	17,191,123,611	18,098,370,199
br_immed_spec	114,327,927,453	75,846,301,505	77,748,996,109	81,077,538,494	74,363,986,924	81,816,217,422
br_indirect_spec	4,766,921,930	4,023,537,222	2,408,015,927	3,132,816,002	2,297,597,243	3,307,499,079
br_return_spec	4,177,843,578	3,414,625,897	2,007,795,372	2,568,481,934	1,912,352,958	2,696,994,072

Table 4.24: Branch Prediction and Speculation Benchmarks for Miller Rabin Method 2 - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
br_pred	2,159,899,097	4,329,960,998	210,846,242	661,706,466	2,196,271,385	4,187,899,602	319,533,673	552,967,080	2,088,395,405	4,155,425,105	252,182,651	448,612,038
branch-misses	453,646,795	911,893,785	31,561,740	89,433,544	509,330,461	1,040,541,570	60,119,120	101,237,441	517,084,710	1,030,932,033	38,914,238	73,671,443
br_immed_spec	2,129,063,831	4,269,502,950	284,078,859	454,946,818	2,143,561,496	4,172,922,541	192,208,142	609,299,294	2,084,594,897	4,145,872,675	194,327,130	450,462,076
br_indirect_spec	33,315,686	67,611,324	14,839,008	24,003,265	51,265,856	25,172,205	8,771,695	30,852,563	12,191,862	23,950,120	10,136,819	22,915,670
br_return_spec	30,627,349	62,073,176	14,130,761	19,791,923	36,352,279	19,731,983	6,848,379	23,419,969	9,446,308	19,013,366	6,666,114	18,269,878

for the 5000-prime set, multithreading slashed the values by approximately 90% and 85% for O2 and O3, respectively, when compared to their standard counterparts. For the 10000-prime set, this decrease hovered around 85% and 89%. This pronounced difference suggests a superior level of parallelism and load balancing for Method 2 when multithreading is applied. Transitioning to the branch-misses benchmark, both methods showed a decline in values when implementing software optimizations. However, multithreading introduced a slight increase in misses, potentially due to inherent complexities in managing concurrent threads, leading to more cache and subsequent branch misses. As for the br_immed_spec benchmark, Method 1 (Table 4.23) witnessed reductions of about 32% with multithreading and approximately 32-35% with O2 and O3. Method 2 (Table 4.24), while presenting a more varied range, generally emphasized the benefits of software optimizations—conversely, the benchmarks br_indirect_spec and br_return_spec between the two tables relay mixed outcomes. While the application of software optimizations often resulted in lowered values, multithreading occasionally prompted an increase, particularly evident in Method 2. This suggests potential complexities in data synchronization among threads. In summation, while different optimizations can enhance specific performance metrics, their overall effectiveness depends on variables such as the number of primes used and the specific benchmark under consideration.

4.3.2 Optimizations with Transparent Huge Pages

Comparing Table 4.17 and Table 4.25 using the Miller Rabin Method 1, using Transparent Huge Pages (THP) resulted in significant performance improvements. There was a decrease in context switches by 34.5% for Normal and 29.3% for multithreading in O3 optimization. While page faults remained relatively stable with less than 1% increase, cycles and instructions, saw reductions of up to 82% and 72%, respectively. Time taken in seconds also improved, with a notable 194% decrease in multithreading. Furthermore, cache misses decreased by up to 58%, and memory accesses – both read and write – witnessed sharp declines, hitting reductions as high as 89%.

For Table 5.18 compared to Table 5.26 using the Miller Rabin Method 2, in the scenario of 10,000 test cases with O3 optimization, THP again demonstrated its efficacy. Context switches showed a minor decrease of 4% in Normal, though a slight 0.7% increase was observed in Multithreading. Almost negligible changes were recorded for page faults. However, substantial performance boosts were noted in cycles and instructions, which decreased by as much as 95%—the execution time reflected these improvements, showing up to a 53% reduction. Cache misses and memory accesses, including both read and write operations, followed suit with impressive decreases, some reaching up to 99%. In essence, Transparent Huge Pages consistently amplify performance across both Miller Rabin methods, especially in domains of memory access and computational efficiency.

In assessing the performance of Miller Rabin Method 1, especially when observing the benchmarks related to L1 cache operations, significant variances are evident between the original software optimizations and those supplemented by Transparent Huge Pages

Table 4.25: General Benchmarks for Miller Rabin Method 1 with Transparent Huge Pages - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
context-switches	38,036	53,348	38,969	139,599	26,124	137,428
page-faults	112	128	111	129	111	129
cycles	801,415,471,466	682,979,501,518	811,401,111,613	747,202,445,903	543,355,505,115	998,479,588,748
instructions	769,750,307,523	657,165,845,442	495,008,975,481	464,982,913,717	331,420,604,567	619,984,173,266
seconds	448.771512054	145.926679092	454.261878638	116.124890797	304.202897183	300.615189048
cache-misses	27,497,558	122,991,846	27,621,297	265,361,096	18,163,729	241,715,649
mem_access	456,125,728,176	384,726,320,353	20,929,266,041	30,660,877,182	13,918,635,501	35,281,605,698
mem_access_rd	314,596,551,912	264,012,043,242	12,554,827,311	16,957,337,643	8,382,837,546	19,894,145,581
mem_access_wr	141,571,774,485	121,402,790,202	8,378,350,136	13,724,101,017	5,541,363,448	15,325,885,646

Table 4.26: General Benchmarks for Miller Rabin Method 2 with Transparent Huge Pages- Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
context-switches	356	738	12,578	23,878	177	325	10,338	20,694	13,763	362	10,245	20,943
page-faults	111	111	10,348	20,583	110	110	10,350	20,585	110	111	10,350	20,584
cycles	24,475,957,815	48,796,696,488	2,320,072,247	5,552,962,598	14,939,042,550	30,095,160,513	1,462,236,209	3,307,623,565	15,690,173,985	29,738,693,681	2,045,217,506	3,180,829,771
instructions	23,071,307,192	45,985,979,085	2,071,544,012	5,246,665,380	8,360,363,433	16,783,354,291	860,464,865	1,906,605,836	8,957,798,433	16,625,315,241	1,148,226,535	1,926,927,516
seconds	13.701670706	27.303683040	5.314270466	10.566518030	8.361924610	16.838648175	4.043084725	7.925624347	9.350139207	16.668726039	4.113783852	7.930297707
cache-misses	1,571,966	3,011,054	3,158,998	5,777,605	1,206,536	2,446,730	2,362,807	6,787,557	6,034,990	2,344,258	2,908,290	6,647,332
mem_access	13,796,192,857	27,513,994,477	703,368,773	2,492,902,636	105,100,942	211,336,620	108,043,299	290,481,158	341,386,479	206,955,313	126,837,212	264,174,891
mem_access_rd	9,541,115,027	19,015,862,941	796,015,370	1,893,856,771	53,351,920	107,443,665	48,447,062	105,127,397	179,835,689	110,444,155	49,967,171	101,275,842
mem_access_wr	4,261,463,800	8,493,524,074	351,863,654	915,597,452	49,656,895	101,591,184	56,944,010	113,082,115	154,689,193	103,294,957	53,020,583	97,241,717

Table 4.27: L1 Cache Benchmarks for Miller Rabin Method 1 with Transparent Huge Pages - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
L1-dcache-loads	456,117,733,505	384,811,957,787	20,931,549,494	30,671,394,201	13,923,247,486	35,321,085,968
l1d_cache_refill	27,326,957	124,157,199	28,140,537	264,158,010	18,349,849	240,554,497
l1d_cache_rd	314,553,432,474	264,183,388,025	12,572,428,450	16,973,782,008	8,395,425,192	19,902,251,657
l1d_cache_wr	141,573,868,437	121,401,081,752	8,377,613,864	13,796,122,561	5,538,449,327	15,308,231,694
l1d_cache_wb	2,703,234	50,579,312	2,936,009	103,465,037	1,713,754	93,277,027
l1d_tlb_refill	13,199,680	86,995,417	13,334,633	122,533,995	9,089,646	107,924,522
l1d_cache_inval	109,942	48,298,137	104,680	98,121,134	71,447	87,163,269
L1-icache-loads	596,935,681,994	502,206,682,768	520,884,191,055	471,721,692,295	342,695,302,223	602,824,105,119
l1i_cache_refill	50,633,351	71,081,791	49,971,781	130,888,904	33,150,279	129,360,879
l1i_tlb_refill	1,532,376	16,233,624	1,623,572	23,332,053	1,023,058	18,282,233

Table 4.28: L1 Cache Benchmarks for Miller Rabin Method 2 with Transparent Huge Pages - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
L1-dcache-loads	13,799,678,064	27,510,904,718	2,032,592,798	2,104,960,925	102,411,097	210,435,347	157,850,031	266,254,325	334,916,839	210,767,497	132,713,068	322,656,329
l1d_cache_refill	1,607,729	3,021,576	4,386,300	4,600,926	1,186,792	2,463,203	3,762,621	5,780,379	5,841,843	2,328,838	2,468,271	6,107,889
l1d_cache_rd	9,543,469,653	19,016,037,482	819,743,254	1,786,294,585	53,440,416	110,115,005	57,094,295	126,777,296	173,431,070	108,956,705	50,005,527	92,055,954
l1d_cache_wr	4,260,456,215	8,493,929,529	421,787,293	817,321,683	51,165,194	99,469,615	34,708,038	112,821,372	156,623,345	104,721,724	56,484,285	103,846,849
l1d_cache_wb	348,648	682,869	2,371,796	2,716,878	297,135	672,671	1,805,265	3,354,134	1,413,847	590,313	1,111,698	3,437,641
l1d_tlb_refill	1,388,201	2,749,123	1,757,854	1,847,812	1,238,585	2,401,940	1,127,797	2,348,881	3,724,958	2,504,219	763,853	2,697,273
l1d_cache_inval	156,970	309,478	74,507	194,438	162,828	323,737	77,429	164,499	476,135	321,490	62,014	112,690
L1-icache-loads	18,361,135,779	36,604,158,028	2,324,310,355	2,841,455,356	9,659,707,397	19,455,160,495	898,627,665	2,109,792,785	9,649,836,145	18,556,159,960	713,441,311	1,980,428,751
l1i_cache_refill	4,496,338	9,205,977	2,876,928	5,770,926	3,438,550	6,681,273	2,462,793	6,039,763	15,903,027	6,838,426	2,913,887	5,929,699
l1i_tlb_refill	241,156	501,897	120,032	224,380	168,759	343,835	95,178	284,652	506,172	398,350	120,614	297,613

(THP). For the L1-dcache-loads under normal conditions, Table 4.27 with THP shows a decrease by approximately 35% compared to Table 4.19. In a multithreading context, the decrease is more subtle, nearly 15%. However, under O2 optimization, the L1-dcache-loads with THP improve by about 55% in a normal context and by around

35% in a multithreaded scenario. For O3 optimization, the improvement with THP is roughly 8% under normal conditions and a significant 64% in a multithreading scenario. The l1d_cache_refill metric in the normal setting for Table 4.27 is reduced by about 36% compared to Table 4.19. However, with multithreading, the values in Table 4.27 are reduced by around 14%. Under O2 optimization, the l1d_cache_refill values are increased by about 51% in normal and by a substantial 95% in multithreading with THP. For O3 optimization, there is a 3% increase in the normal context and a whopping 35% increase in multithreading with THP.

Comparatively, Upon analyzing the L1 cache benchmarks for Miller Rabin Method 2 between Table 4.20 (software optimizations) and Table 4.28 (with THP), it is evident that Transparent Huge Pages (THP) generally augments the performance across different optimizations and conditions. For instance, when comparing L1-dcache-loads for the 5000 value under normal conditions, there is a marked improvement of about 27% in Table 4.28 as opposed to Table 4.20. This trend is more profound for the 10000 value under normal conditions, witnessing a remarkable improvement of approximately 38%. In the realm of multithreading, the figures for 5000 and 10000 showcase an improvement of 18% and 21%, respectively, with the application of THP. The l1d_cache_refill, in a normal context for the value 5000, reveals an elevation of roughly 31% in performance in Table 4.28. For 10000, the rise is even more prominent at about 40%. Multithreading scenarios for 5000 and 10000 show an increase of 23% and 17%, respectively. Similar improvement patterns with the introduction of THP are visible in other benchmarks, such as l1d_cache_rd, l1d_cache_wr, and l1d_cache_wb in both normal and multithreading contexts. In conclusion, the Transparent Huge Pages seem to enhance performance in Miller Rabin Methods 1 and 2 across various benchmarks and scenarios. The results underscore the efficacy of THP in optimizing cache operations, especially in computational-intensive tasks. However, specific benchmarks did note a decrease or negligible change with THP, suggesting a nuanced approach in its application.

Tables 4.21 and 4.29 shed light on the L2 Cache Benchmarks for the Miller Rabin Method 1, emphasizing software optimizations and the utilization of Transparent Huge Pages (THP), respectively. In Table 4.21, the O3 optimization level with multithreading stands out with an l2d_cache (475,963,615). This represents an impressive increase of approximately 550% when compared to the standard setting (73,144,080). However, when we evaluate the incorporation of THP in Table 4.29 for the same method, the l2d_cache count dips slightly to 660,802,038 for the multithreaded scenario under O3, a decline of roughly 13.2% from the non-THP equivalent.

Table 4.29: L2 Cache Benchmarks for Miller Rabin Method 1 with Transparent Huge Pages - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
l2d_cache	113,578,854	363,807,352	112,789,023	714,829,711	75,039,637	660,802,038
l2d_cache_refill	2,137,160	1,373,135	344,986	2,460,847	245,697	2,423,992
l2d_cache_rd	85,075,384	249,091,669	85,267,087	479,174,505	56,716,755	439,583,914
l2d_cache_wr	28,095,308	118,557,706	28,410,876	238,589,984	18,712,727	216,882,213
l2d_cache_wb	347,652	188,609	60,021	347,148	34,878	340,629
l2d_cache_inval	0	0	0	0	0	0

Table 4.30: L2 Cache Benchmarks for Miller Rabin Method 2 with Transparent Huge Pages - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
l2d_cache	8,621,354	16,368,451	8,535,116	27,135,208	6,443,039	13,393,481	11,922,773	29,851,345	31,215,326	12,580,112	14,553,244	25,863,414
l2d_cache_refill	255,304	291,259	1,369,136	5,350,841	130,910	301,178	2,328,008	6,238,423	334,963	200,042	2,809,402	4,969,565
l2d_cache_rd	6,612,064	13,005,481	8,985,623	18,814,666	5,228,460	10,931,723	8,615,517	14,530,211	23,974,425	10,591,158	9,532,042	15,447,251
l2d_cache_wr	1,705,338	3,181,185	3,458,200	7,434,230	1,207,022	2,538,982	2,937,152	6,617,597	6,295,452	2,420,440	3,625,179	7,013,864
l2d_cache_wb	29,187	39,270	523,691	2,183,210	17,335	43,053	988,846	2,616,676	37,543	28,402	1,156,364	2,090,281
l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Tables 4.22 and 4.30 delve into the L2 Cache Benchmarks for Miller Rabin Method 2. The multithreaded count under O3 for 10000 iterations in both tables is prominent, showing l2d_cache (counts of 22,430,695 and 25,863,414) in Tables 4.22 and 4.30, respectively. This indicates an increase of about 15.3% when THP is applied. Analyzing the l2d_cache_refill metric, which points to cache misses, Table 4.22's multithreaded setup under O2 optimization for 10000 iterations reveals 4,898,604 refills. On the other hand, in Table 4.30, with THP in play, this number escalates to 6,238,423 under similar conditions, marking an increase of approximately 27.4% in cache misses. In summary, applying software optimizations and Transparent Huge Pages can markedly sway L2 cache performance metrics. The effects fluctuate based on the method deployed and specific workload conditions, including iteration count and multithreading. While THP can elevate the L2 cache hit count by around 15.3% in some scenarios, it can also induce a surge of 27.4% in cache misses in others. It is crucial to conduct a thorough system examination when planning to optimize performance, as indicated by these percentages.

Finally, the branch prediction and speculation. For the br_pred benchmark in Table 4.23 versus Table 4.31, there is an evident reduction of approximately 35% in normal mode and around 14% in multithreading mode. However, when O2 optimizations are applied, the trend inverts, showing an increase of about 55% for normal mode and 38% for multithreading mode in Table 4.31. For the O3 optimizations, while there is a slight

Table 4.31: Branch Prediction and Speculation Benchmarks for Miller Rabin Method 1 with Transparent Huge Pages - Software Optimizations

Benchmarks			**O2**		**O3**	
	Normal	Multithreading	Normal	Multithreading	Normal	Multithreading
br_pred	77,188,700,416	68,610,920,000	124,258,770,047	115,861,838,285	83,190,643,315	154,918,920,950
branch-misses	14,671,207,436	12,172,980,880	27,842,627,042	24,480,968,137	18,661,839,997	33,495,523,570
br_immed_spec	74,086,305,662	65,035,529,309	120,554,113,793	111,342,203,496	80,714,542,785	149,403,874,953
br_indirect_spec	3,114,331,089	3,503,868,960	3,732,757,994	4,651,557,859	2,490,886,253	5,606,050,054
br_return_spec	2,731,888,044	2,967,433,383	3,113,355,663	3,794,947,057	2,072,976,695	4,597,170,231

Table 4.32: Branch Prediction and Speculation Benchmarks for Miller Rabin Method 2 with Transparent Huge Pages - Software Optimizations

Benchmarks					**O2**				**O3**			
	Normal		**Multithreading**		**Normal**		**Multithreading**		**Normal**		**Multithreading**	
	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000	5000	10000
br_pred	2,172,018,598	4,320,432,602	273,372,566	602,588,634	2,080,272,311	4,194,692,167	280,923,876	482,850,420	2,239,761,191	4,152,392,904	238,935,155	534,714,585
branch-misses	456,146,478	909,464,433	37,623,398	90,498,572	516,350,834	1,042,014,808	41,212,255	80,556,670	520,206,858	1,031,282,931	50,546,897	89,279,511
br_immed_spec	2,134,107,342	4,255,037,390	264,176,469	424,706,767	2,074,231,394	4,181,091,477	227,702,020	406,664,710	2,195,913,412	4,149,377,275	241,173,600	513,281,016
br_indirect_spec	33,775,003	67,020,706	16,885,569	26,243,797	12,595,997	24,640,232	11,499,129	22,425,273	55,213,932	23,985,595	13,666,794	25,418,838
br_return_spec	30,779,320	61,620,055	13,495,764	21,650,417	9,928,317	19,321,870	9,002,736	17,560,988	38,605,151	18,822,197	11,581,066	16,980,403

dip of 10% for normal mode, multithreading mode sees a significant jump of 82%. Shifting the focus to the branch misses benchmark, the normal mode of operation in Table 4.31 demonstrates a drop by 35% and multithreading by 15% when compared to Table 4.23. On the other hand, with O2 optimization, Table 4.31 showcases an upward trend with an increase of 55% for normal mode and 36% for multithreading mode. Under O3 optimizations, normal mode increases by 9%, and multithreading mode rises considerably by 85%. Furthermore, consistent patterns can be observed in benchmarks like br_immed_spec, br_indirect_spec, and br_return_spec between these two tables, aligning with the abovementioned trends.

For Method 2, comparing the br_pred benchmarks between Table 4.24 and Table 4.32 shows varied results. In normal mode for inputs 5000 and 10000, there's a 0.6% increase and a 0.2% decrease, respectively. Multithreading mode sees a 30% rise for input 5000 but a 27% drop for 10000. O2 optimizations give mixed results, reducing normal and multithreading modes. O3 optimizations boost the 5000 input by 7% in normal mode but cause drops in other scenarios. Similar trends appear in benchmarks, like branch misses. In summary, using Transparent Huge Pages with the Miller Rabin methods results in improvements and declines, showcasing the complex relationship between software tweaks, modes, and memory techniques.

Chapter 5

Discussion and Conclusion

Cryptographic optimization heavily relies on memory management efficiency, with Transparent Huge Pages (THP) offering significant benefits when paired with specific encryption modes like ECB. ECB mode's independent block encryption and decryption facilitate parallel processing when combined with THP, enhancing throughput and efficiency. However, ECB's susceptibility to revealing data patterns due to constant ciphertext for identical plaintext presents security risks, rendering it unsuitable for most applications despite its performance benefits. CBC and CFB modes lack parallel decryption due to block interdependence, yet pipelining technique circumvents this constraint. Here, decryption dependencies in CBC, or block cipher encryption in CFB, allow for parallel operations. Despite THP's benefits, the partial parallelization in these modes introduces more complex control flow and extra memory needs, potentially offsetting benefits, especially with smaller datasets. Notwithstanding, encryption in both modes is inherently parallelizable, further enhancing THP's cache and branch prediction efficiencies. Regarding Miller-Rabin methods, Method 1 significantly reduces branch misses through software optimization techniques like O2 and O3. Despite increased total branch predictions, Method 2 minimizes branch misses via efficient multithreading, indicating effective flow control. When THP is paired with Miller-Rabin, the larger memory pages reduce the frequency of TLB misses. Given that the Miller-Rabin method involves iterative calculations, any improvement in memory retrieval and storage, courtesy of THP, can result in substantial performance gains. This enhancement is especially noticeable when Miller-Rabin is utilized for testing larger prime numbers, where the computational requirements are more intensive and memory access patterns more recurrent. Additionally, the synergistic application of THP with software optimizations, as seen with O2 and O3, and multithreading, further

leverages the inherent parallelism potential of the Miller-Rabin method. The larger contiguous memory blocks provided by THP ensure that data required for concurrent threads is swiftly accessible, reducing latency and aiding in maintaining a consistent computational rhythm. This is particularly beneficial for Method 2 of Miller-Rabin, where the efficient use of multithreading minimizes branch misses and maximizes data access efficiency. In essence, THP can really speed up cryptographic tasks, especially when paired with ECB. But we've got to choose the correct encryption method and consider its strengths and weaknesses. With Miller-Rabin's tests, combining THP with good software practices shows how much faster and more efficient things can get. As we move forward, understanding how to use memory best and manage complexity will be vital to making more progress.

In conclusion, the demonstrated benefits of Transparent Huge Pages (THP) significantly optimized memory utilization during cryptographic processes on Raspberry Pi 4. Through multithreading and OpenMP, we harnessed the Pi's hardware to reveal considerable performance improvements, particularly when utilizing parallelizable ECB mode. Nonetheless, ECB's known security shortcomings necessitate cautious application. Furthermore, while we have introduced a degree of parallelism into modes like CBC and CFB via pipelining, we must acknowledge the intricacies it introduces into control flow and its potential to increase memory demands, which could impact performance gains. In the Miller-Rabin analysis, Method 1 showed impressive branch-miss reduction via O2 and O3 optimization techniques. In contrast, Method 2, despite a rise in total branch predictions, exhibited efficient multithreading for superior flow control. In terms of future work, scaling up this study to a cluster of Raspberry Pi devices would be an intriguing path to investigate. The possibility of exploiting the parallel processing capabilities of encryption modes across multiple devices and prime number generation presents a promising opportunity for improving efficiency and throughput. Based on the findings and insights from this research, future work could enhance the cryptographic operations within a clustered environment, thereby harnessing the full potential of these cost-effective and powerful computational devices. By furthering this line of inquiry, we can continue to provide valuable insights into optimizing computational workloads, ultimately contributing to the ongoing efforts to enhance the performance of these increasingly essential systems. In sum, our work here represents an encouraging step forward in utilizing Raspberry Pi devices for complex cryptographic operations more efficiently.

In conclusion, the demonstrated benefits of Transparent Huge Pages (THP) signifi-

cantly optimized memory utilization during cryptographic processes on Raspberry Pi 4. Through multithreading and OpenMP, we harnessed the Pi's hardware to reveal considerable performance improvements, particularly when utilizing parallelizable ECB mode. Nonetheless, ECB's known security shortcomings necessitate cautious application. Furthermore, while we have introduced a degree of parallelism into modes like CBC and CFB via pipelining, we must acknowledge the intricacies it introduces into control flow and its potential to increase memory demands, which could impact performance gains. In the Miller-Rabin analysis, Method 1 showed impressive branch-miss reduction via O2 and O3 optimization techniques. In contrast, Method 2, despite a rise in total branch predictions, exhibited efficient multithreading for superior flow control. In terms of future work, scaling up this study to a cluster of Raspberry Pi devices would be an intriguing path to investigate. The possibility of exploiting the parallel processing capabilities of encryption modes across multiple devices and prime number generation presents a promising opportunity for improving efficiency and throughput. Based on the findings and insights from this research, future work could enhance the cryptographic operations within a clustered environment, thereby harnessing the full potential of these cost-effective and powerful computational devices. By furthering this line of inquiry, we can continue to provide valuable insights into optimizing computational workloads, ultimately contributing to the ongoing efforts to enhance the performance of these increasingly essential systems. In sum, our work here represents an encouraging step forward in utilizing Raspberry Pi devices for complex cryptographic operations more efficiently.

Bibliography

[1] Israel Tekahun Basha and Meron Istifanos. Performance evaluation of raspberry pi 3b as a web server : Evaluating the performance of raspberry pi 3b as a web server using nginx and apache2, 2020.

[2] Jonathan Corbet. Transparent hugepages, Oct 2009.

[3] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[4] Parikshit Dubey and Arva Kagdi. Run time analysis of matrix multiplication using raspberry pi cluster supercomputer. 05 2020.

[5] Raspberry Pi Foundation. Raspberry pi, Jun 2023.

[6] Wikimeadia Foundation. Block cipher mode of operation, Jul 2023.

[7] Karl Fürlinger. Openmp application profiling — state of the art and directions for the future. *Procedia Computer Science*, 1(1):2107–2114, 2010.

[8] Scott Gilbertson. The raspberry pi 4 can't quite replace a pc, but it tries, Aug 2019.

[9] Wajdi Hajji and Fung Po Tso. Understanding the performance of low power raspberry pi cloud for big data. *Electronics*, 5(2), 2016.

[10] Gareth Halfacree and Eben Upton. *Raspberry Pi user guide*. John Wiley & Sons, 2012.

[11] Branislav Madoš, Ján Hurtuk, Eva Chovancová, Peter Fecil'ák, and Dávid Bajkó. Downsizing of web server design using raspberry pi 3 single board computer platform. In *2017 IEEE 14th International Scientific Conference on Informatics*, pages 238–242, 2017.

[12] Garry Miller and Michael Rabbin. Miller–rabin primality test, May 2023.

[13] Anand Nayyar and Vikram Puri. Raspberry pi-a small, powerful, cost effective and efficient form factor computer: A review. *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, 5:720–737, 12 2015.

[14] National Institute of Standards and NIST Technology. Advanced encryption standard (aes), May 2023.

[15] Kernel Org. Perf, 2009.

[16] The Regents of the University of California. Pthreads overview (for lc) - uml.edu, Feb 2000.

Appendix A

Perf Events

There are several types of events we will focus on, grouped into four categories:

- **Context switches, page faults, cycles, instructions, time elapsed, cache misses, Memory Access**: These events are associated with how efficiently the program runs. Context switches and page faults could indicate unnecessary overhead. Monitoring cycles, instructions, and elapsed time can help us understand where the program spends its time. Cache misses and memory access patterns can highlight opportunities for optimizing memory usage.

- **L1 Cache Usage**: L1 cache events such as L1-dcache-loads, l1d_cache_refill, l1d_cache_rd, l1d_cache_wr, l1d_cache_wb, l1d_tlb_refill, and l1d_cache_inval help us understand how effectively our program is using the L1 data cache. Similarly, L1-icache-loads, l1i_cache_refill, and l1i_tlb_refill give insights into the L1 instruction cache usage. Optimizing for cache usage can significantly speed up a program.

- **L2 Cache Usage**: L2 cache events such as l2d_cache, l2d_cache_refill, l2d_cache_rd, l2d_cache_wr, l2d_cache_wb, and l2d_cache_inval shed light on the L2 cache utilization. These metrics can help detect potential bottlenecks in memory access and areas where cache usage can be improved.

- **Branch Predictions**: Branch prediction events, such as br_pred and branch-misses, give us insights into the efficiency of the CPU's branch prediction mechanism. Good branch prediction is crucial for maintaining high instruction throughput.

Each of these metrics will provide valuable insights into the operation and performance of the code. The ultimate goal is to identify potential areas of improvement to optimize the execution time and efficiency of the code under investigation.

This systematic approach of employing readily available resources from GitHub and utilizing the Linux perf tool for performance analysis provides a robust framework for optimizing and improving code performance, all within the accessible and compact package of the Raspberry Pi 4.

Appendix B

optimizations

B.1 Multithreading and OpenMP

Parallelization techniques, such as multithreading and OpenMP, are pivotal for leveraging the Raspberry Pi 4's multiple cores:

Both of these techniques exploit the multiple cores in the Raspberry Pi 4's CPU to run code in parallel, but they differ in their abstraction and ease of use.

OpenMP [7] (Open Multi-Processing) is a popular parallel programming model for shared-memory multiprocessors. Many compilers support it and provide a high-level and scalable approach to parallelization. OpenMP is implemented using compiler directives, which are special instructions you insert into your code. These directives instruct the compiler to execute certain parts of the code in parallel.

An important advantage of OpenMP is its simplicity. To parallelize a code section, you often need to add a single pragma — a special type of comment that the compiler interprets as an instruction. For example, to parallelize a for loop, you might write something like this:

```
#pragma omp parallel for
for (int i = 0; i < N; i++) {
  // ... loop body ...
}
```

The compiler then automatically creates multiple threads and divides the iterations of the loop among them. This allows the developer to focus more on the logic of their code rather than the details of parallelization. However, OpenMP also provides various options for more advanced control over parallelization, such as setting the number of

threads, scheduling the distribution of iterations among threads, and controlling the scope of variables. This flexibility makes OpenMP a powerful tool for optimizing code to use the available hardware resources efficiently.

In contrast, multithreading involves manually creating and managing multiple threads of execution within a program. This is a lower-level approach compared to OpenMP and requires a more thorough understanding of parallel programming concepts. However, it also offers greater control and flexibility. The most common method of implementing multithreading in C++ is by using the POSIX threads[1] [16] (pthreads) library. This involves using function calls provided by the library to create threads, specify the function each thread should run, and manage the threads. Here is an example of how you might create two threads:

```
#include <pthread.h>

void* my_function(void* arg) {
  // ... code to be run in a separate thread ...
}

int main() {
  pthread_t thread1, thread2;

  pthread_create(&thread1, NULL, my_function, NULL);
  pthread_create(&thread2, NULL, my_function, NULL);

  pthread_join(thread1, NULL);
  pthread_join(thread2, NULL);

  return 0;
}
```

This code, pthread_create, is used to create each thread and start it running my_function. pthread_join is then used to wait for each thread to finish before the program exits. multithreading allows for more sophisticated parallelization strategies, such as using different threads to perform different tasks concurrently or creating a dynamic number of threads based on the run-time conditions of the program. However, it also introduces addi-

[1] https://man7.org/linux/man-pages/man7/pthreads.7.html

tional complexity, such as the need to handle synchronization between threads and the potential for race conditions.

By employing these parallelization techniques, we aim to accelerate the processing times of the codes. We expect that the OpenMP approach will be more suitable for easily parallelizable, compute-intensive portions of the code, while the multithreaded approach will provide finer control for optimizing more complex, interaction-intensive sections. It should be noted that using these techniques requires careful consideration of the trade-off between the overhead of creating and managing multiple threads and the potential for speedup through parallel execution. The optimal solution will likely involve a combination of these techniques tailored to the specific characteristics of the code.

Our primary aim in this exploration is to optimize the performance of our cryptographic implementations. The most crucial aspect we are targeting is minimizing execution time; achieving a shorter time for executing programs is a significant victory in computing, as it increases the overall efficiency and throughput. However, in parallel to lowering execution time, we also examine the inner workings of memory access and usage to optimize for lower translation look-aside buffer (TLB) and cache misses. These misses represent instances where data is not in the processors' cache or TLB and must be fetched from a slower memory layer or disk, causing delay. This is important because an improved understanding of memory usage leads to more effective utilization of the cache and TLB, drastically improving performance. When discussing utilizing the cache as much as possible, we refer to our effort to keep the most frequently or recently used data in the cache closer to the processor. This is because accessing data from the cache is significantly faster than getting it from the main memory or, even worse, the disk. By structuring our program to make optimal use of the cache, we can speed up the overall execution. As we proceed with our optimizations, we will focus on direct optimizations, like changing the code to reduce computational overhead, and indirect ones, like ensuring optimal memory usage. However, it is essential to understand that improving performance is not just about optimizing the code. It is also about understanding what is happening beneath the surface. This is where tools like perf come into play, providing detailed statistics on various performance aspects such as cache and TLB misses, branch predictions, and many others. These statistics allow us to make educated assumptions about the system's behavior and guide our optimization strategies. This quest to understand what these results mean extends beyond mere curiosity. Based on these results, it allows us to draw valuable conclusions about the

workload. For instance, high cache miss rates suggest the need for optimizing data locality or rethinking the algorithms and data structures in use. High rates of TLB misses suggest that we must look into our applications' page size or memory access patterns. High-branch mispredictions could be a signal to look at the structure of our conditional statements and control flow.

B.2 Compiler Optimizations

Compiler-level optimizations, specifically the O2 and O3 flags, harness the power of the compiler to refine and improve the generated code's performance:

The O2 optimization flag is a compiler directive that activates a set of optimizations aimed at enhancing the execution speed of the compiled program without unduly increasing its size. These optimizations might include, for instance, loop unrolling, constant propagation, and dead-code elimination. The O2 level, therefore, offers a balanced approach, ensuring efficient execution without a significant increase in binary size. On the other hand, the O3 optimization flag is a step further in this optimization journey. It activates a broader range of aggressive optimizations, including those under O2 and additional tactics that might significantly increase the binary size, like function inlining and vectorization. The aim is to squeeze out every bit of performance, even if it means the compiled program will occupy more space.

It is crucial to understand that these optimization flags don't modify the original code but instruct the compiler on generating the machine code. This means that while the source code remains unchanged, the binary produced can differ greatly in terms of its structure and performance. The choice between O2 and O3 (or even lower optimization levels) often boils down to the specific needs of the application and the platform it's running on. Some applications might benefit more from the aggressive optimizations under O3, while others might find the balance struck by O2 more suitable. It's also worth noting that while these optimizations generally improve performance, they can occasionally introduce subtle bugs or issues, especially when using O3. Hence, thorough testing is advised after compiling these flags. Furthermore, while these optimizations improve code execution speed, they should ideally be used in tandem with other optimization strategies. Techniques like multithreading and OpenMP, which leverage hardware capabilities like multiple cores, can significantly enhance the effects of compiler optimizations. As we proceed with our exploration, we will consider compiler-level optimizations and more manual, code-level techniques to achieve the

best possible performance on the Raspberry Pi 4 system. This comprehensive approach ensures that we address performance bottlenecks from multiple angles, maximizing the potential of our implementations.

Appendix C

Tables - ECB

Table C.1: General Benchmarks for ECB - Software Optimizations

	Benchmarks	128 Bits Key			192 Bits Key			256 Bits Key		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	context-switches	21,367	11,445	11,160	24,998	15,394	12,948	28,702	19,124	17,849
	page-faults	121,177	121,194	135,834	121,178	121,194	134,786	121,178	121,192	128,935
	cycles	448,182,508,025	464,915,822,743	454,234,208,076	523,062,215,405	548,044,065,193	534,612,086,062	602,541,850,714	634,059,935,557	621,873,389,204
	instructions	821,488,496,616	814,155,599,117	818,640,835,905	971,898,162,941	965,370,819,340	971,276,900,654	1,123,087,941,695	1,119,958,797,119	1,136,960,763,811
	time elapsedd (s)	250.882695938	284.32801	256.86591	292.821577181	126.333165559	110.127100761	52,289,636	155.143214603	138.480971156
	cache-misses	292,567,688	52,584,773	68,324,031	48,843,823	56,442,601	68,361,383	52,289,636	60,661,726	73,691,828
	mem_access	346,013,028,058	347,476,356,499	342,692,861,892	409,491,330,862	413,789,561,044	405,767,403,238	472,304,431,263	478,799,154,548	473,733,041,368
	mem_access_rd	273,431,573,054	277,162,503,349	269,994,897,303	324,990,227,067	328,650,425,738	321,696,526,903	376,885,312,204	371,965,018,395	365,448,924,637
	mem_access_wr	72,860,027,571	70,313,853,150	71,752,840,876	84,124,695,423	85,588,350,012	83,022,995,759	95,362,031,528	95,110,390,850	92,501,373,711
ECB 676 MB	context-switches	41,590	28,562	31,914	50,375	36,760	37,595	54,547	44,531	44,258
	page-faults	242,245	242,252	285,431	242,236	242,250	273,512	242,234	242,251	264,090
	cycles	893,028,495,730	939,106,603,007	912,824,870,164	1,050,331,037,227	1,098,138,588,911	1,072,930,690,475	1,205,931,356,779	1,268,756,873,631	1,235,442,959,996
	instructions	1,642,416,199,787	1,638,369,843,790	1,641,102,229,100	1,944,946,377,621	1,934,470,239,914	1,949,324,434,167	2,249,274,578,019	2,235,069,251,045	2,254,631,506,731
	time elapsed (s)	500.074510864	235.423360468	263.190373700	588.127485352	296.301782968	306.693249652	675.025688597	349.361548273	351.218127447
	cache-misses	118,927,583	111,899,748	136,375,830	111,580,890	121,043,966	162,829,991	108,658,169	135,297,815	181,090,256
	mem_access	696,237,533,922	696,609,637,406	680,745,830,392	820,495,734,317	823,521,821,573	809,740,586,581	944,360,389,112	952,071,104,503	936,203,667,541
	mem_access_rd	549,277,601,709	549,660,514,195	540,422,077,284	652,011,896,689	656,797,392,693	648,720,567,767	753,578,665,398	758,587,641,803	747,183,486,762
	mem_access_wr	146,874,415,762	147,919,750,137	143,877,627,319	168,949,784,615	171,129,323,043	167,091,894,559	190,874,089,947	193,648,213,174	188,211,110,164
ECB 1135 MB	context-switches	85,199	53,467	62,879	101,640	71,522	74,584	116,246	85,102	87,219
	page-faults	484,342	484,357	572,874	484,344	484,357	529,302	484,341	484,357	511,290
	cycles	1,785,188,812,464	1,879,447,018,524	1,813,996,414,993	2,094,090,602,874	2,185,597,375,979	2,126,932,992,390	2,422,024,438,645	2,536,560,387,355	2,455,872,267,116
	instructions	3,284,428,228,780	3,290,810,245,724	3,267,917,966,485	3,891,288,989,780	3,843,362,578,446	3,863,422,217,126	4,497,236,159,580	4,490,301,061,262	4,481,708,000,521
	time elapsed (s)	999.320867307	452.571035518	517.575247984	1172.608764791	584.687547987	601.272257815	1355.838643633	681.941710454	685.701113597
	cache-misses	178,206,742	203,763,709	300,443,117	205,980,448	263,827,045	278,036,350	253,253,966	236,983,487	330,463,518
	mem_access	1,391,549,895,617	1,397,874,431,263	1,371,533,001,981	1,639,801,554,027	1,654,535,401,019	1,632,274,880,651	1,890,986,227,084	1,901,599,371,099	1,862,886,922,268
	mem_access_rd	1,098,058,974,526	1,107,147,468,166	1,094,210,898,304	1,302,043,227,246	1,299,951,998,344	1,267,402,311,296	1,508,662,759,789	1,517,646,294,903	1,491,877,956,716
	mem_access_wr	293,519,151,725	297,791,527,761	291,054,903,558	337,531,821,723	339,415,182,514	327,750,834,788	382,061,508,160	386,692,890,334	375,318,828,825

Table C.2: General Benchmarks for ECB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	context-switches	7,228	3,030	3,119	3,050	2,320	2,327	7,027	3,303	3,451	4,188	2,314	2,374	6,717	3,739	3,526	3,887	2,604	2,517
	page-faults	121,179	121,195	250,190	121,178	121,192	278,480	121,178	121,193	241,498	121,178	121,195	274,734	121,178	121,192	256,047	121,180	121,193	266,502
	cycles	110,441,889,083	109,050,240,622	109,833,753,780	66,475,594,577	65,535,451,154	66,835,352,721	125,860,337,195	124,427,555,048	126,128,279,808	73,798,437,200	73,611,050,697	75,077,555,273	142,682,634,897	142,159,101,868	140,957,534,552	81,615,068,371	81,406,782,029	81,002,619,423
	instructions	216,521,700,204	212,899,222,383	212,155,605,352	119,802,924,148	119,327,442,879	118,919,964,354	247,993,540,206	243,681,454,781	245,969,393,049	133,309,815,313	133,453,656,261	134,601,554,843	280,130,906,013	278,955,889,062	277,091,457,148	148,217,916,301	147,808,250,623	142,817,298,715
	time elapsed (s)	62.372056448	28.411071992	28.701468338	37.252561183	22.318277670	22.542876292	70.538357743	30.641306405	30.787587216	41.383972722	23.317995237	23.558933607	79.900445352	32.988503751	33.064546579	45.719792177	24.358153349	24.721736912
	cache-misses	34,844,625	33,005,157	50,948,562	24,092,448	29,223,108	44,736,838	29,533,375	36,406,773	46,818,122	38,786,719	27,525,073	52,905,674	32,688,667	28,850,760	50,348,042	27,654,673	25,378,853	118,501,650
	mem_access	79,599,751,005	79,078,350,778	79,926,663,359	58,563,624,763	58,339,028,481	57,271,904,496	90,877,542,623	90,583,907,184	91,848,213,635	65,762,481,161	65,572,897,448	63,980,159,996	102,685,385,712	101,288,498,258	102,560,951,180	72,922,341,565	71,739,177,363	71,383,071,735
	mem_access_rd	54,687,694,622	53,677,443,428	53,242,792,284	39,170,357,715	38,477,137,943	38,232,940,292	62,123,509,766	61,148,025,931	61,212,668,309	44,216,956,035	44,016,310,736	44,634,423,041	70,302,310,478	70,438,112,976	69,474,033,773	49,295,653,349	49,105,111,163	49,359,391,158
	mem_access_wr	25,718,571,688	25,228,678,539	25,181,768,945	18,938,295,225	18,764,852,442	18,403,569,598	28,561,927,325	28,099,834,018	28,265,253,220	20,890,272,468	20,503,713,601	21,317,030,878	31,782,047,347	31,840,813,647	31,556,731,660	22,913,136,630	22,878,557,275	23,213,878,033
ECB 676 MB	context-switches	15,743	9,941	6,050	8,163	5,965	4,409	11,798	6,727	7,206	8,316	4,668	4,913	15,541	7,138	7,312	9,254	4,835	5,017
	page-faults	243,194	242,249	502,846	246,668	242,250	559,961	242,234	242,250	482,726	243,871	242,249	551,860	242,236	242,250	490,835	242,235	242,248	548,508
	cycles	220,887,325,627	221,048,984,215	220,615,721,056	133,635,655,716	219,161,144,452	134,864,689,330	252,689,593,261	250,805,189,507	251,953,489,374	148,278,613,518	147,198,697,358	150,987,094,641	284,368,919,807	283,121,780,643	283,907,873,496	163,193,566,687	162,706,418,306	163,877,118,129
	instructions	433,056,236,537	431,223,378,583	426,472,086,711	241,916,088,237	428,048,149,138	239,060,124,467	494,445,023,051	491,096,650,516	491,388,886,538	268,951,192,075	267,576,400,100	269,535,420,960	559,552,543,109	557,360,198,751	555,850,992,805	296,348,342,506	293,944,930,122	290,967,606,028
	time elapsed (s)	124.193796408	58.706254634	57.249411930	74.976730200	56.736496594	45.054691822	141.508625333	61.281016147	61.577485391	83.115246962	46.580457550	47.039593348	159.378395745	65.596175343	66.201362137	91.482186109	48.718367771	49.38021 1918
	cache-misses	73,976,904	59,992,015	98,225,882	63,114,176	57,096,019	100,270,061	59,569,517	86,339,645	128,149,738	55,754,628	53,004,593	94,316,006	68,214,092	57,376,554	96,392,213	56,441,922	84,041,224	96,667,867
	mem_access	159,742,123,220	159,194,654,331	159,616,130,295	117,138,398,102	159,062,352,455	116,144,807,013	182,309,100,871	181,598,164,304	182,411,559,125	131,330,703,081	130,573,354,461	129,874,156,591	205,359,929,260	205,014,751,289	204,580,585,244	145,394,101,020	144,421,387,250	145,657,264,763
	mem_access_rd	109,072,906,125	108,552,760,282	107,306,344,365	78,724,399,419	108,047,969,534	79,523,726,699	124,429,518,520	123,806,566,016	124,215,502,640	88,716,907,043	88,307,752,638	89,410,288,065	140,703,155,593	140,352,525,596	140,649,738,987	99,256,816,336	98,721,655,101	98,042,833,147
	mem_access_wr	51,326,907,331	51,176,864,799	50,707,520,882	37,961,815,117	50,789,637,886	38,685,138,311	57,225,015,832	56,750,428,553	57,265,072,162	41,930,566,771	41,686,555,894	42,570,176,699	63,606,733,972	63,479,519,216	63,952,167,894	46,044,180,534	45,673,276,055	45,893,677,759
ECB 1135 MB	context-switches	28,806	12,632	11,858	20,079	11,961	8,912	24,014	13,049	13,618	14,621	9,374	9,305	27,519	14,150	14,287	15,384	9,923	9,820
	page-faults	501,073	496,004	1,064,242	511,878	500,882	1,133,697	485,755	484,357	1,040,581	485,927	484,358	1,110,931	485,969	484,401	942,531	484,342	484,357	1,102,323
	cycles	442,580,481,693	438,296,396,586	443,246,274,142	270,751,361,175	438,782,549,653	271,193,709,685	503,356,955,485	503,302,875,320	507,149,506,457	294,491,302,963	295,782,308,813	300,447,401,942	569,913,491,830	566,280,493,686	572,109,197,248	323,094,812,409	326,002,210,571	330,193,954,626
	instructions	863,934,107,490	859,113,178,778	863,885,961,494	489,132,135,125	858,319,231,974	482,879,955,260	988,666,969,018	987,386,266,509	993,402,094,664	537,838,868,045	537,621,729,107	537,491,136,349	1,118,588,786,669	1,116,761,016,235	1,121,975,003,359	591,379,416,364	589,690,684,783	588,271,254,590
	time elapsed (s)	249.563045045	113.369047909	113.796999197	153.628169407	113.222074580	89.949902321	281.882087788	122.270555457	122.933512471	164.963336982	92.942068784	93.931715683	319.134837236	131.089955845	131.852282209	181.004039965	97.844667868	98.127300402
	cache-misses	118,853,158	101,192,475	188,296,202	104,189,286	105,830,302	202,522,767	127,624,690	126,084,524	240,775,204	124,422,113	114,204,141	190,294,468	125,886,411	147,999,784	212,668,179	118,705,167	126,672,129	210,878,371
	mem_access	320,104,346,146	318,101,847,132	321,444,946,208	236,175,300,029	318,593,479,770	236,784,374,985	364,504,200,572	363,636,267,790	366,960,347,390	262,463,523,447	262,656,951,463	264,672,697,949	410,196,305,460	410,034,269,975	411,083,649,410	291,576,659,404	290,903,167,061	292,009,039,036
	mem_access_rd	217,739,676,739	216,132,912,943	217,207,695,700	158,231,542,890	216,182,845,545	157,572,693,064	249,455,101,402	249,059,955,116	249,825,359,631	178,084,474,156	177,400,119,103	179,045,924,600	281,891,669,764	281,588,938,171	282,487,556,730	198,467,854,066	197,677,430,333	199,980,875,941
	mem_access_wr	102,540,716,346	101,552,315,750	102,619,411,281	76,792,773,882	101,481,652,476	76,198,345,008	114,687,475,844	114,559,392,910	115,443,565,126	84,129,069,724	83,706,137,968	85,119,620,108	127,505,936,786	127,288,081,000	128,106,337,853	92,164,969,825	91,899,639,278	93,172,909,970

Table C.3: General Benchmarks for ECB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**			**192 Bits Key**			**256 Bits Key**		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	context-switches	23,440	10,530	13,159	23,583	13,484	16,751	30,504	17,110	19,637
	page-faults	4,970	4,867	5,134	4,820	5,188	5,222	5,187	4,915	4,843
	cycles	447,735,983,937	467,893,824,406	431,350,990,899	499,905,631,497	549,238,104,912	524,445,471,403	609,085,929,053	636,805,919,593	598,591,402,318
	instructions	820,142,809,228	819,058,833,361	786,594,387,653	924,445,187,266	967,737,348,765	965,856,184,688	1,123,877,453,163	1,124,889,785,800	1,111,290,166,538
	time elapsed (s)	251.002815301	91.158393820	109.909251287	279.999970813	109.337434015	137.030537988	341.388934029	136.284585948	159.258333051
	cache-misses	43,695,107	40,573,568	58,139,454	55,842,670	43,556,381	57,257,750	45,273,252	41,013,637	67,136,702
	mem_access	348,993,238,168	350,224,667,881	343,707,255,878	390,887,567,786	411,494,252,532	395,974,326,787	472,477,598,196	469,078,101,976	456,326,677,253
	mem_access_rd	275,347,614,186	275,918,290,033	269,349,042,032	309,956,427,529	326,842,573,932	323,621,389,528	376,826,832,543	379,984,701,892	371,381,834,932
	mem_access_wr	73,590,264,285	73,966,208,671	70,474,646,855	80,780,769,755	84,959,475,699	81,951,023,992	95,514,064,038	96,673,862,713	91,985,851,879
ECB 676 MB	context-switches	46,210	27,757	24,432	50,154	34,707	27,529	59,279	40,663	31,244
	page-faults	50,799	5,851	6,025	13,193	5,739	5,791	5,336	6,059	6,006
	cycles	898,329,091,747	950,879,033,939	894,335,225,243	1,052,245,166,015	1,107,423,328,454	1,052,281,371,999	1,208,090,243,012	1,279,395,353,642	1,208,153,993,875
	instructions	1,640,802,445,588	1,662,937,989,406	1,587,531,934,619	1,942,069,815,697	1,949,803,323,852	1,881,584,865,333	2,245,466,468,203	2,262,466,174,495	2,173,303,563,286
	time elapsed (s)	503.517910244	231.030110442	177.218940432	588.989438472	281.248887027	199.914174571	677.253538471	330.154270336	221.410983923
	cache-misses	83,969,088	129,839,470	95,103,035	182,319,563	94,360,661	153,904,814	87,939,453	119,002,965	109,426,028
	mem_access	698,603,237,425	696,748,994,106	682,844,669,084	820,614,247,517	820,835,661,032	804,995,085,737	946,952,797,320	942,611,481,488	928,293,357,062
	mem_access_rd	551,200,824,470	552,371,920,404	541,102,410,959	652,440,779,489	656,203,172,786	642,468,328,804	754,923,445,059	766,119,872,981	742,693,263,260
	mem_access_wr	147,570,229,261	148,337,911,482	142,206,251,245	168,977,261,372	170,727,088,897	163,729,950,618	191,026,637,838	194,915,314,341	184,778,392,865
ECB 1135 MB	context-switches	93,216	42,558	40,977	103,521	48,659	46,749	119,715	55,035	53,052
	page-faults	7,093	6,679	6,990	6,951	6,790	7,215	6,777	6,664	7,163
	cycles	1,781,119,216,898	1,872,903,596,953	1,786,526,317,287	2,096,945,605,100	2,204,415,503,869	2,104,025,496,979	2,408,607,234,736	2,536,007,544,064	2,420,027,559,503
	instructions	3,281,122,628,691	3,274,325,466,035	3,177,894,236,872	3,885,182,634,727	3,881,232,960,385	3,761,337,565,134	4,491,183,601,407	4,485,780,707,316	4,351,895,185,324
	time elapsed (s)	998.905450503	364.705022429	353.055019867	1173.946105507	410.680805699	399.083311392	1348.462099066	456.850222890	441.763620519
	cache-misses	1,131,542,692	134,359,704	219,210,843	166,000,132	180,659,849	249,305,273	199,083,278	145,588,064	1,194,602,590
	mem_access	1,384,363,242,260	1,399,998,898,687	1,368,107,007,080	1,639,520,255,077	1,650,330,341,734	1,614,134,692,107	1,888,059,670,796	1,901,368,530,722	1,857,172,064,052
	mem_access_rd	1,093,751,636,523	1,103,595,124,296	1,083,609,600,057	1,303,197,404,459	1,308,838,757,954	1,284,949,533,978	1,506,721,093,752	1,515,271,461,570	1,488,092,881,756
	mem_access_wr	291,454,043,924	296,009,724,656	284,487,620,943	337,340,513,748	340,553,773,987	327,105,874,423	381,205,217,190	385,474,434,474	370,231,794,321

Table C.4: General Benchmarks for ECB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	context-switches	7,192	3,064	3,276	2,965	2,141	2,239	6,304	3,263	3,385	3,519	2,265	2,347	6,675	3,849	3,701	3,869	2,438	2,516
	page-faults	5,475	4,805	6,429	4,884	4,756	5,651	4,740	5,188	5,681	5,444	4,692	5,400	5,668	5,650	5,353	4,691	4,819	6,945
	cycles	109,212,654,082	108,447,821,909	107,927,782,954	65,140,814,145	64,728,737,068	64,586,339,412	124,625,028,531	123,616,916,402	123,650,655,844	72,226,973,275	72,576,573,032	72,611,706,513	140,496,290,559	140,733,751,392	139,641,138,671	79,810,414,994	80,167,531,477	78,680,033,145
	instructions	212,504,628,147	211,851,720,068	209,170,185,226	119,411,445,529	119,076,600,553	112,759,878,670	246,507,837,760	243,018,405,764	241,789,111,295	133,072,049,480	132,212,498,379	129,589,034,213	278,587,693,591	277,551,952,811	273,487,410,888	146,204,186,781	145,182,018,752	138,214,426,726
	time elapsed (s)	61.715505775	28.081570739	28.020081564	36.515729581	21.802773320	22.108817343	69.793742526	30.119477413	30.285347919	40.498316675	22.859071431	23.058674517	78.668983721	32.606163104	32.483307029	44.704459181	23.948011504	24.092003188
	cache-misses	23,051,566	21,407,804	47,269,732	28,121,957	23,644,515	94,884,853	25,738,707	20,222,009	45,162,140	30,750,850	19,402,308	46,029,132	24,349,980	25,669,214	48,582,441	26,852,606	34,795,024	39,626,246
	mem_access	79,512,429,960	77,580,145,258	77,618,309,023	57,808,336,858	57,395,310,054	57,054,005,534	90,626,811,698	90,379,691,398	89,472,832,436	65,523,897,222	63,947,153,502	63,287,610,586	101,869,321,940	100,987,409,481	102,017,381,830	72,478,634,726	70,991,926,541	71,092,625,572
	mem_access_rd	54,291,444,412	53,929,203,951	53,640,173,294	39,430,027,974	38,704,386,436	37,596,738,681	62,272,951,505	61,344,026,948	60,357,969,172	44,268,493,217	43,731,498,430	43,552,564,754	70,457,091,512	70,218,036,576	68,731,600,968	49,540,380,418	48,899,273,837	48,808,140,173
	mem_access_wr	25,418,760,540	25,162,788,399	24,870,305,964	18,717,672,518	18,635,601,757	18,039,914,304	28,530,060,344	28,067,887,337	27,501,516,244	20,853,507,504	20,130,196,516	20,475,012,962	31,675,318,054	31,569,065,592	30,855,533,811	22,741,307,043	22,385,822,045	22,219,083,889
ECB 676 MB	context-switches	14,330	6,268	6,311	6,335	4,598	4,504	12,003	6,516	6,345	7,043	4,654	4,661	13,667	7,169	7,322	7,603	5,137	4,976
	page-faults	6,043	5,818	7,868	9,903	5,996	7,849	5,869	5,933	7,715	5,965	5,705	7,293	6,080	5,802	7,069	5,821	5,739	8,355
	cycles	218,381,531,620	217,211,419,869	217,332,462,757	131,065,585,117	129,942,677,314	131,691,535,121	248,547,132,779	248,879,247,701	246,975,075,421	144,465,902,707	145,103,409,842	144,452,741,234	281,265,712,188	280,107,778,289	282,118,751,054	159,018,409,965	159,058,760,708	160,451,606,026
	instructions	427,040,853,729	424,952,636,232	421,461,504,263	239,342,386,375	236,966,308,928	233,329,588,923	490,433,596,562	488,945,827,349	484,951,393,678	265,786,673,894	263,906,246,954	257,442,856,134	557,216,004,180	553,721,651,397	552,973,447,393	292,272,883,119	289,497,971,286	285,788,585,818
	time elapsed (s)	123.149168803	55.693205434	55.935829438	73.413959779	43.528532259	44.302103566	139.214154573	60.390945867	60.261786171	80.953621379	45.710084442	46.011727220	157.496893338	64.805393458	65.025106021	89.072688984	47.686842681	48.026871632
	cache-misses	48,829,277	51,074,024	89,360,167	48,864,494	35,184,085	167,351,110	48,232,507	46,352,260	97,401,876	50,325,154	59,614,242	87,077,464	66,122,424	63,039,743	110,484,028	50,749,364	36,420,307	86,772,751
	mem_access	159,468,327,103	157,658,233,836	158,302,098,627	115,298,925,151	114,881,310,397	114,081,493,402	181,477,582,489	181,034,173,658	178,800,455,768	129,446,671,700	129,897,595,906	128,499,888,480	203,577,403,169	203,470,955,811	201,875,539,635	144,306,690,561	143,863,187,640	143,455,243,341
	mem_access_rd	107,851,429,500	107,917,059,159	106,777,338,387	78,814,890,861	77,671,714,351	77,844,647,327	124,014,100,193	123,498,312,817	124,050,395,018	88,572,960,583	88,273,178,362	88,149,173,155	140,824,105,197	140,367,421,156	140,201,922,352	98,705,983,282	97,935,232,219	97,563,679,471
	mem_access_wr	50,508,231,561	50,241,514,009	49,731,850,853	37,614,248,418	37,302,254,232	37,015,107,284	56,675,687,601	56,419,314,110	56,590,249,437	41,421,695,653	41,347,361,388	41,095,704,783	63,404,300,579	63,115,056,810	62,890,192,834	45,573,020,306	45,389,000,319	44,829,778,870
ECB 1135 MB	context-switches	29,325	12,098	12,161	20,235	8,829	9,069	22,539	13,078	13,415	13,760	9,120	9,402	27,362	14,336	14,542	15,098	9,685	9,751
	page-faults	140,143	6,999	9,321	135,499	6,919	9,016	6,881	7,110	9,391	7,047	6,807	8,113	7,015	6,663	8,509	6,647	6,871	9,167
	cycles	437,721,811,914	434,172,476,846	438,560,286,251	264,168,022,631	261,879,974,852	265,962,066,612	497,142,815,975	497,441,443,956	501,575,322,989	288,711,146,935	290,826,888,694	301,283,521,147	561,632,528,375	561,948,853,256	565,761,527,320	317,902,624,172	320,672,045,635	324,538,391,279
	instructions	858,621,957,900	852,764,231,494	850,079,344,032	482,367,735,329	476,181,028,449	472,093,829,571	982,023,842,198	980,141,041,724	982,831,884,312	530,808,691,194	528,708,996,357	523,022,677,181	1,113,405,505,976	1,111,225,800,817	1,109,266,369,022	584,135,525,442	581,437,183,307	576,663,881,490
	time elapsed (s)	246.560812814	111.730813038	112.389771929	150.864755935	87.071005522	88.457155492	278.421531707	120.209068737	121.077746840	161.733504745	91.102709487	93.360069796	314.475843845	129.636074523	130.454146409	178.040679370	95.358108773	96.138199424
	cache-misses	99,532,951	78,540,904	173,300,769	95,535,265	80,821,076	342,215,237	84,461,029	81,127,486	247,301,040	80,449,365	92,734,262	847,542,771	131,182,805	88,718,074	178,777,595	93,533,424	79,486,470	195,953,689
	mem_access	318,909,430,922	315,900,523,367	315,086,324,527	233,944,712,194	230,157,066,836	228,255,768,761	362,829,334,392	361,163,692,853	360,641,035,603	259,995,558,586	259,971,511,257	259,487,061,209	407,036,390,274	407,181,582,269	408,099,317,948	288,957,351,086	288,455,888,891	285,185,959,020
	mem_access_rd	216,624,125,633	215,078,170,174	215,497,837,215	157,987,205,038	156,736,377,330	157,567,362,095	248,008,937,142	247,998,017,410	248,104,652,796	177,419,927,942	176,282,964,946	176,095,538,178	281,278,872,745	280,354,837,847	280,278,798,152	197,598,381,574	196,414,615,862	196,373,830,091
	mem_access_wr	101,500,171,962	100,426,579,259	100,364,396,069	76,075,720,362	74,472,270,679	75,132,247,905	113,360,696,547	113,286,464,678	113,018,205,978	83,148,719,239	82,765,687,891	81,899,141,437	126,453,912,869	126,217,732,249	125,918,282,693	91,095,609,775	90,543,315,197	89,911,374,285

Table C.5: L1 Cache Benchmarks for ECB - Software Optimizations

	Benchmarks	128 Bits Key			192 Bits Key			256 Bits Key		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	346,089,844,708	347,705,299,523	342,741,174,886	409,602,882,520	411,117,649,107	404,189,146,595	472,571,592,047	478,507,053,355	473,626,164,448
	l1d_cache_refill	291,126,361	52,266,627	68,001,500	49,185,984	55,799,585	64,539,353	52,434,218	60,611,149	70,382,295
	l1d_cache_rd	273,316,517,286	277,162,503,349	269,994,897,303	325,224,818,294	329,009,704,455	320,217,037,982	376,509,878,087	375,971,335,221	357,566,019,196
	l1d_cache_wr	72,914,939,765	74,331,075,019	71,752,840,876	84,270,058,519	85,711,466,524	82,491,328,479	95,448,338,486	96,178,550,129	89,971,630,866
	l1d_cache_wb	15,196,353	19,150,868	26,258,040	15,947,917	13,834,726	28,502,394	14,096,063	16,284,619	27,300,144
	l1d_tlb_refill	18,415,562	19,820,832	17,341,183	19,912,558	22,403,942	20,590,123	19,179,740	25,103,077	22,647,425
	l1d_cache_inval	38,648	340,895	7,385,135	51,003	464,733	8,310,308	50,051	524,078	5,779,199
	L1-icache-loads	260,213,797,468	266,036,296,935	250,257,340,798	310,724,524,965	318,519,006,985	294,598,048,968	359,231,099,763	366,132,417,144	343,818,168,724
	l1i_cache_refill	63,593,716	56,042,079	44,079,768	70,290,283	64,939,124	52,176,652	73,330,554	72,583,912	61,547,292
	l1i_tlb_refill	1,787,039	1,709,717	1,684,939	2,012,153	2,040,547	1,685,477	2,260,430	1,953,270	1,864,696
ECB 676MB	L1-dcache-loads	696,126,379,099	700,261,642,576	677,266,762,671	821,010,509,850	814,205,735,769	810,692,613,711	944,052,196,753	951,256,845,678	928,143,898,483
	l1d_cache_refill	122,655,024	111,223,100	144,832,705	110,356,534	118,187,588	164,380,972	109,705,437	134,847,013	181,235,411
	l1d_cache_rd	548,969,263,853	555,154,983,106	543,403,837,879	652,070,991,355	655,013,158,068	647,353,877,371	753,568,763,217	762,381,502,724	740,250,261,797
	l1d_cache_wr	146,820,710,240	149,118,592,590	144,917,619,866	169,086,170,854	171,739,135,340	167,218,842,151	190,890,355,687	194,715,970,920	187,631,361,666
	l1d_cache_wb	39,593,312	27,443,026	56,637,312	34,165,121	27,782,961	53,187,143	35,704,747	34,349,461	56,420,922
	l1d_tlb_refill	38,883,262	39,768,353	47,837,013	51,238,088	47,264,424	54,517,714	46,202,599	54,081,834	58,543,364
	l1d_cache_inval	134,774	833,680	14,229,135	77,697	1,129,223	12,389,742	206,279	1,408,453	9,991,491
	L1-icache-loads	530,484,752,669	539,557,892,814	492,952,042,563	626,153,652,410	628,280,266,828	589,669,364,017	717,994,815,000	729,427,271,366	676,380,929,483
	l1i_cache_refill	129,231,470	127,576,804	124,436,887	143,903,372	148,075,689	139,788,890	149,179,572	164,775,316	152,784,608
	l1i_tlb_refill	4,203,592	3,684,402	3,697,366	4,071,959	4,487,512	4,401,588	3,948,161	4,746,516	4,599,649
ECB 1351MB	L1-dcache-loads	1,392,050,542,636	1,398,146,449,694	1,367,999,817,190	1,639,222,671,240	1,645,343,959,503	1,640,655,441,957	1,890,663,921,763	1,898,810,288,831	1,864,823,509,571
	l1d_cache_refill	175,483,727	200,122,631	301,042,145	209,070,021	257,087,595	279,033,356	251,585,262	235,215,166	331,085,419
	l1d_cache_rd	1,098,028,899,163	1,103,393,985,739	1,097,562,809,400	1,302,292,153,550	1,319,346,953,291	1,288,506,988,822	1,507,975,782,363	1,515,468,830,097	1,501,980,012,458
	l1d_cache_wr	293,380,663,726	296,956,680,650	292,113,513,477	337,618,200,424	343,698,909,125	331,925,054,984	382,222,207,517	386,718,861,569	378,310,831,586
	l1d_cache_wb	54,904,638	60,988,477	109,244,755	69,805,733	69,651,274	93,497,867	64,637,389	56,565,474	102,974,396
	l1d_tlb_refill	71,232,264	82,271,565	91,802,581	90,105,358	97,765,205	98,493,392	83,536,760	120,096,326	112,883,081
	l1d_cache_inval	182,967	1,714,384	30,152,189	138,262	2,100,939	16,880,970	171,213	2,573,024	12,377,814
	L1-icache-loads	1,059,741,091,594	1,072,999,138,726	1,000,517,013,393	1,245,526,245,622	1,263,599,996,514	1,195,883,435,555	1,443,715,430,869	1,457,363,078,036	1,360,778,656,651
	l1i_cache_refill	259,244,651	244,600,757	242,577,071	288,999,759	287,293,589	261,170,398	302,702,520	309,029,804	287,037,932
	l1i_tlb_refill	7,556,563	7,434,377	7,593,546	8,756,839	8,659,393	8,039,220	9,337,472	8,412,223	8,881,558

Table C.6: L1 Cache Benchmarks for ECB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3				O2			O3	
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	80,451,744,909	79,021,919,996	79,305,065,806	58,418,295,290	58,441,653,821	58,676,371,434	91,419,313,875	90,502,234,729	91,276,570,664	65,858,176,030	65,230,149,770	63,310,889,071	102,786,651,840	101,715,684,412	102,110,946,250	73,061,964,891	72,378,892,097	71,693,795,249
	l1d_cache_refill	34,827,013	32,116,493	49,559,550	27,679,493	28,564,582	43,745,035	27,814,245	36,011,863	45,975,821	41,751,299	26,461,415	50,752,763	31,749,761	28,341,640	48,825,798	26,801,938	24,902,010	117,352,356
	l1d_cache_rd	54,364,835,537	53,793,373,213	52,979,465,119	39,654,463,095	38,291,381,734	37,846,540,448	62,389,609,843	62,324,535,716	61,230,169,923	44,536,647,928	43,751,778,065	43,985,691,100	70,353,988,667	70,336,815,156	69,141,351,512	49,304,626,807	49,122,489,337	49,005,410,378
	l1d_cache_wr	25,539,510,804	25,312,067,089	25,087,460,154	19,091,205,715	18,477,618,780	18,202,677,561	28,740,699,758	28,611,431,180	28,146,421,320	21,009,510,973	20,895,672,335	21,073,074,379	31,986,899,253	31,817,869,815	31,587,341,269	23,026,123,136	22,894,420,470	23,105,076,297
	l1d_cache_wb	14,782,960	14,733,351	27,380,823	13,710,380	13,553,172	25,909,221	11,860,535	12,296,479	26,614,202	17,502,162	12,690,785	28,315,442	12,490,045	12,028,256	27,737,771	11,920,590	12,014,885	46,186,858
	l1d_tlb_refill	10,537,988	9,930,772	18,550,945	10,936,366	9,754,337	20,071,156	10,542,597	10,748,400	17,666,885	9,386,636	10,569,354	19,537,967	10,829,965	10,604,238	19,222,191	9,797,759	10,175,511	18,828,926
	l1d_cache_inval	11,370	139,584	11,562,165	7,170	127,152	12,196,521	44,644	156,868	11,673,286	26,327	113,907	12,158,366	15,994	144,719	11,952,304	6,984	118,571	12,214,268
	L1-icache-loads	72,911,231,479	72,146,852,222	72,746,009,814	39,248,746,845	38,752,757,529	39,127,095,633	83,075,115,416	82,510,755,506	83,875,503,826	42,792,628,963	42,344,771,322	41,419,197,243	94,404,311,636	93,031,128,685	93,302,014,346	46,473,470,474	45,827,867,004	45,394,059,040
	l1i_cache_refill	28,119,249	19,881,601	24,728,454	18,709,848	17,288,945	26,015,325	26,562,641	20,984,257	25,457,244	20,349,343	17,979,676	25,006,964	26,757,576	22,154,366	28,805,352	20,213,146	18,540,890	25,562,914
	l1i_tlb_refill	742,169	499,135	472,930	464,582	506,387	608,644	456,536	402,995	658,849	477,428	468,837	580,631	519,111	514,553	678,726	521,371	363,551	598,461
ECB 676MB	L1-dcache-loads	160,781,681,258	159,816,140,880	159,458,883,085	117,105,954,440	159,324,126,869	117,720,093,524	182,420,407,101	181,504,385,181	182,086,046,429	131,489,436,311	130,421,635,130	131,178,576,889	205,294,314,545	204,546,168,606	204,510,718,624	145,924,125,416	145,450,221,751	145,230,555,441
	l1d_cache_refill	67,988,040	60,690,745	96,104,993	62,430,602	56,450,823	96,217,826	58,376,476	85,885,212	127,121,455	54,161,062	51,630,239	94,503,777	67,145,385	55,798,500	94,748,352	54,789,349	83,345,970	98,700,914
	l1d_cache_rd	108,540,772,352	108,264,699,756	108,391,760,863	79,017,453,529	107,765,161,686	79,082,578,690	124,935,422,383	124,503,893,732	125,028,233,376	89,476,441,641	88,184,463,288	89,214,242,537	140,990,977,146	140,510,032,695	141,191,515,952	99,367,730,899	98,890,456,260	99,093,953,067
	l1d_cache_wr	51,219,458,877	50,940,009,858	51,144,483,815	38,005,002,802	50,657,992,762	38,171,468,402	57,433,475,546	57,193,292,878	57,329,300,773	42,125,069,613	41,787,008,579	42,512,417,000	63,818,625,574	63,571,333,282	64,056,016,940	46,193,703,430	45,929,599,980	46,713,083,256
	l1d_cache_wb	28,576,213	25,681,102	56,506,454	29,336,974	26,186,660	57,302,802	30,582,799	34,023,143	61,347,778	28,660,173	25,721,740	55,537,905	34,023,005	23,132,276	52,126,328	25,062,030	40,369,250	57,414,264
	l1d_tlb_refill	24,577,242	23,550,640	37,624,964	23,475,050	24,486,287	40,987,226	23,033,915	23,515,145	38,532,836	20,843,184	23,113,891	40,779,304	23,882,656	23,820,057	36,657,299	21,623,256	21,282,790	38,632,541
	l1d_cache_inval	77,740	262,900	23,881,188	63,791	277,348	25,864,416	23,211	313,732	22,698,176	51,653	235,116	25,088,209	98,235	313,262	22,696,350	79,643	243,973	24,447,857
	L1-icache-loads	145,328,666,629	145,678,848,434	145,814,787,667	78,170,596,351	144,822,364,963	78,574,272,298	167,137,672,335	165,554,357,451	166,347,493,954	85,412,271,727	84,750,784,848	84,378,784,041	187,550,372,684	186,567,134,469	186,834,262,319	92,614,065,602	92,553,195,536	92,022,849,245
	l1i_cache_refill	61,090,618	50,029,287	50,096,534	42,483,795	42,994,494	51,525,735	50,646,394	43,655,879	53,703,071	40,939,660	37,695,330	52,894,347	57,158,721	47,059,461	55,222,558	43,695,033	37,977,319	51,903,320
	l1i_tlb_refill	1,223,295	1,247,466	1,104,721	856,659	1,027,444	3,612,189	1,300,356	1,045,991	1,207,796	1,112,702	732,785	1,141,370	1,299,440	758,874	978,163	954,546	1,129,945	1,087,336
ECB 1351MB	L1-dcache-loads	320,745,997,736	318,294,588,676	321,244,895,700	235,819,546,052	318,642,423,211	235,866,600,929	364,324,805,635	363,741,305,915	366,672,978,273	262,687,885,700	262,562,714,818	264,926,953,979	410,011,671,569	409,672,477,402	411,991,933,364	291,397,771,274	291,256,368,286	292,818,306,321
	l1d_cache_refill	113,408,172	100,047,236	186,885,688	106,341,464	107,582,491	198,598,525	126,003,002	131,600,305	237,800,163	126,369,695	111,918,388	188,164,175	124,464,066	145,514,524	211,855,568	120,805,947	125,760,654	212,306,608
	l1d_cache_rd	217,255,792,813	216,555,141,494	217,906,411,154	158,554,582,544	215,815,343,589	158,686,967,988	249,335,995,614	249,060,587,468	250,415,502,267	178,123,924,949	177,346,220,434	178,882,973,505	281,950,298,325	281,607,270,760	282,245,707,079	198,565,623,194	197,619,787,595	199,802,836,193
	l1d_cache_wr	102,318,045,986	101,469,419,744	102,453,000,262	76,992,646,079	101,506,364,295	77,513,535,114	114,759,833,269	114,590,722,328	115,719,206,192	84,182,312,153	83,610,853,012	84,699,139,549	127,694,467,636	127,475,106,212	128,023,790,494	92,348,135,336	91,783,714,171	93,442,313,175
	l1d_cache_wb	48,078,053	45,318,220	109,287,632	50,870,306	50,837,818	117,372,060	52,547,512	65,346,732	128,978,240	60,488,916	58,181,664	112,698,808	53,578,919	61,613,819	109,753,788	54,181,059	56,240,305	126,680,151
	l1d_tlb_refill	48,040,348	45,880,430	81,584,036	51,907,815	47,985,232	86,023,716	47,729,159	48,232,919	85,647,615	43,460,732	43,170,747	83,858,939	47,172,384	48,427,508	90,876,448	43,650,221	45,371,182	93,448,740
	l1d_cache_inval	62,945	527,213	49,808,194	56,377	550,275	50,709,700	30,942	563,850	47,603,044	34,461	489,692	51,025,770	42,680	603,401	46,298,322	22,764	466,988	49,997,368
	L1-icache-loads	291,571,505,363	288,839,221,693	292,183,503,534	157,252,353,959	289,497,196,801	157,255,041,323	332,173,563,201	331,981,011,362	333,794,205,521	170,302,706,218	170,262,617,829	171,620,583,721	375,989,141,415	373,109,338,413	376,572,314,001	185,109,462,319	184,955,089,804	186,128,644,934
	l1i_cache_refill	116,113,931	82,605,245	108,588,927	95,560,676	83,348,093	104,559,041	101,333,364	88,939,692	113,861,995	78,378,021	73,991,689	107,285,884	107,417,956	91,736,834	109,548,135	81,214,860	75,102,333	107,421,325
	l1i_tlb_refill	3,186,654	2,075,235	1,840,300	1,952,286	1,991,371	2,185,518	2,629,863	1,514,231	2,521,209	2,303,387	2,211,356	2,685,020	2,846,364	1,468,034	2,635,010	2,337,911	2,003,016	2,121,306

Table C.7: L1 Cache Benchmarks for ECB with Transparent Huge Tables - Software Optimizations

	Benchmarks	128 Bits Key			192 Bits Key			256 Bits Key		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	348,746,356,467	350,074,206,387	335,441,045,688	390,818,917,170	411,060,921,559	399,178,450,999	472,448,292,546	472,125,093,505	460,769,252,688
	l1d_cache_refill	43,330,084	39,514,120	60,412,472	55,724,851	44,080,636	61,457,665	45,205,119	40,375,044	69,132,981
	l1d_cache_rd	275,393,255,069	275,539,505,134	266,658,244,657	309,497,339,073	327,939,871,257	310,982,415,574	376,685,069,057	380,223,686,364	364,929,871,912
	l1d_cache_wr	73,511,067,256	73,771,769,624	69,530,530,795	80,715,506,404	85,191,017,014	79,437,933,014	95,312,078,933	96,587,814,139	89,984,829,839
	l1d_cache_wb	14,129,456	12,756,664	23,326,985	16,613,119	13,487,029	19,646,634	11,741,787	10,021,313	20,062,070
	l1d_tlb_refill	14,210,659	10,999,489	13,374,645	13,101,726	12,479,232	15,062,352	15,364,073	15,749,884	17,360,964
	l1d_cache_inval	33,938	179,369	2,544,602	117,556	295,701	1,850,008	59,006	360,407	1,264,641
	L1-icache-loads	267,207,291,106	269,615,449,764	249,712,088,427	298,932,056,806	315,563,879,675	291,898,526,272	359,632,289,614	360,057,857,483	334,572,578,100
	l1i_cache_refill	40,183,702	26,113,146	32,566,765	40,124,512	31,544,785	38,749,255	48,985,535	38,683,721	44,127,146
	l1i_tlb_refill	1,100,388	1,931,612	1,185,562	1,135,982	2,353,924	1,441,677	1,270,526	2,702,383	1,529,233
ECB 676MB	L1-dcache-loads	698,272,562,651	700,024,819,898	683,660,905,378	820,293,475,428	828,124,830,538	806,775,954,894	947,198,856,423	952,836,847,439	928,597,272,423
	l1d_cache_refill	84,721,715	133,141,472	94,631,585	184,156,607	94,975,630	150,830,893	87,537,866	118,156,054	110,482,128
	l1d_cache_rd	550,860,088,291	542,147,310,763	540,954,715,127	652,143,758,255	654,390,082,485	642,699,329,637	756,247,937,160	761,342,700,541	742,581,203,845
	l1d_cache_wr	147,491,425,659	146,000,880,409	142,248,583,790	169,029,097,118	169,924,298,302	163,709,380,537	191,562,948,150	194,278,127,287	184,692,808,235
	l1d_cache_wb	28,204,011	23,689,481	40,495,365	34,108,926	26,449,605	42,437,493	29,041,668	30,996,959	42,195,957
	l1d_tlb_refill	25,908,778	26,788,326	21,798,319	26,636,513	30,483,890	23,196,166	28,940,792	37,664,104	25,251,587
	l1d_cache_inval	67,710	614,734	10,598,299	70,631	806,028	4,315,012	175,709	1,027,132	4,815,863
	L1-icache-loads	536,627,326,934	537,282,129,569	518,925,869,291	626,367,737,276	633,347,183,441	611,390,459,679	725,367,649,710	730,777,770,334	702,454,152,678
	l1i_cache_refill	95,752,112	63,343,694	55,107,117	82,698,597	78,271,736	62,187,524	90,306,203	91,209,564	66,735,444
	l1i_tlb_refill	2,936,238	5,006,386	1,895,430	2,217,203	5,794,716	2,881,202	2,496,504	6,205,516	2,052,324
ECB 1351MB	L1-dcache-loads	1,384,419,242,170	1,399,391,863,508	1,368,218,713,250	1,639,555,617,254	1,650,156,130,832	1,613,586,697,509	1,888,207,698,892	1,900,640,241,189	1,859,717,828,945
	l1d_cache_refill	1,129,528,740	135,605,929	220,488,828	164,999,533	179,530,229	247,714,259	198,131,289	143,951,254	1,196,166,252
	l1d_cache_rd	1,093,048,758,239	1,103,788,677,840	1,083,807,766,697	1,302,495,821,882	1,309,195,554,112	1,285,464,637,063	1,506,641,854,113	1,515,060,334,460	1,487,843,776,690
	l1d_cache_wr	291,056,831,135	296,280,998,297	284,992,648,256	337,288,716,964	340,656,608,143	327,424,656,818	381,138,542,170	385,427,186,109	370,130,731,554
	l1d_cache_wb	50,354,531	40,501,642	87,320,676	50,217,326	47,781,220	92,010,280	52,056,215	39,875,046	575,299,895
	l1d_tlb_refill	46,755,484	41,798,385	54,556,735	53,166,670	46,877,104	50,143,242	55,695,247	51,168,240	49,572,613
	l1d_cache_inval	125,466	854,226	14,753,660	238,971	1,015,562	8,381,411	368,755	1,325,478	19,095,628
	L1-icache-loads	1,042,985,166,725	1,076,859,208,123	1,036,942,123,314	1,248,841,417,890	1,268,528,943,773	1,221,085,399,124	1,437,945,394,744	1,460,811,793,643	1,406,782,464,356
	l1i_cache_refill	155,275,836	101,294,485	100,878,091	161,350,253	115,250,205	112,381,309	184,519,614	128,030,005	126,878,902
	l1i_tlb_refill	5,057,242	8,018,546	3,775,624	3,902,627	9,545,057	3,787,577	4,413,534	9,313,169	4,337,140

Table C.8: L1 Cache Benchmarks for ECB with Transparent Huge Tables - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	L1-dcache-loads	79,686,778,998	78,746,736,723	75,825,170,405	57,749,403,740	57,270,256,772	56,197,536,385	90,292,049,677	89,767,930,826	89,683,380,699	65,130,030,790	64,221,735,476	61,763,781,349	101,700,738,558	101,819,381,772	100,492,774,528	71,556,106,599	72,617,735,040	70,819,799,737
	l1d_cache_refill	23,535,145	20,565,215	47,417,413	24,097,595	23,184,773	92,991,855	24,904,846	21,341,346	45,221,223	27,895,028	20,282,697	44,138,893	24,552,975	26,117,381	47,338,705	26,913,621	33,327,950	39,381,021
	l1d_cache_rd	54,215,992,913	53,539,546,819	53,574,815,585	39,128,901,144	38,261,055,191	36,569,373,185	61,893,284,152	61,847,179,622	61,082,272,644	44,129,766,134	43,571,304,757	42,377,645,444	70,235,876,070	69,933,265,571	68,491,926,984	49,171,100,340	48,707,085,795	47,946,389,444
	l1d_cache_wr	25,380,352,923	24,938,293,487	25,001,902,828	18,579,628,484	18,476,255,411	18,031,974,176	28,477,632,462	28,119,025,753	27,750,622,140	20,706,426,309	20,633,670,752	20,372,210,186	31,755,927,876	31,374,783,234	30,947,314,204	22,765,444,714	22,330,621,924	22,150,851,264
	l1d_cache_wb	10,139,686	12,189,420	23,274,002	13,625,782	11,515,086	37,086,613	11,860,310	10,074,256	22,101,537	13,800,578	11,458,857	22,659,921	11,134,803	11,813,882	22,572,160	15,342,054	13,853,517	22,270,894
	l1d_tlb_refill	5,896,912	4,874,372	5,158,300	5,171,794	6,052,736	4,230,206	5,491,456	5,151,867	5,633,574	4,710,518	4,871,751	4,284,440	5,695,893	5,598,782	5,241,558	4,626,865	4,538,217	4,730,572
	l1d_cache_inval	5,994	49,904	4,336,984	10,634	17,468	2,952,492	14,460	40,754	3,415,420	8,405	21,395	4,339,451	8,481	72,901	4,546,171	9,018	22,428	4,899,114
	L1-icache-loads	72,300,811,118	71,511,886,971	69,453,104,216	37,980,444,098	37,497,806,370	36,793,306,548	81,948,883,154	81,969,851,390	81,264,477,547	41,465,173,127	41,274,512,016	39,583,386,399	92,419,709,867	92,669,313,830	92,025,640,233	45,219,560,454	44,438,721,369	43,854,593,142
	l1i_cache_refill	15,709,726	8,889,777	8,427,585	8,514,243	6,674,447	6,304,702	11,178,722	8,494,414	8,142,820	8,096,612	7,275,344	6,874,556	11,874,333	10,264,955	8,994,152	8,527,357	6,869,184	6,955,588
	l1i_tlb_refill	494,235	294,712	325,884	271,356	334,092	270,731	355,417	313,650	301,205	247,297	329,992	293,053	351,686	370,921	325,604	303,561	330,086	284,246
ECB 676MB	L1-dcache-loads	159,329,402,865	156,751,577,319	157,735,863,620	115,120,230,370	114,980,838,508	114,585,008,291	180,312,567,719	180,118,245,122	179,548,840,408	129,817,450,548	128,795,241,450	126,453,751,110	203,969,465,267	202,547,906,917	200,810,314,277	143,369,841,305	143,141,787,669	142,798,295,053
	l1d_cache_refill	49,155,459	52,847,170	87,978,047	48,001,944	37,009,683	166,745,997	52,450,613	49,938,564	94,269,926	47,199,658	59,807,612	89,901,823	65,075,120	61,834,464	109,774,461	54,054,360	39,111,924	89,549,980
	l1d_cache_rd	107,971,315,119	107,561,021,296	105,886,857,033	78,369,862,600	77,653,010,797	75,942,985,813	124,377,335,608	123,885,504,518	124,410,421,063	88,250,033,371	87,651,628,507	88,925,898,640	140,201,619,478	139,705,337,893	139,823,172,649	98,871,381,802	98,908,706,781	97,198,679,478
	l1d_cache_wr	50,448,026,352	50,238,020,526	49,469,951,594	37,542,226,487	37,388,980,019	36,110,248,106	56,938,103,094	56,629,330,800	56,761,003,316	41,783,904,769	41,261,256,321	41,280,901,158	63,181,278,791	63,024,538,410	62,889,623,758	45,688,606,061	45,602,239,886	44,297,686,825
	l1d_cache_wb	23,927,565	25,151,583	43,515,116	25,817,671	20,824,261	70,348,462	26,185,431	25,634,962	42,058,008	23,622,188	29,521,293	54,206,139	24,789,246	26,756,845	47,452,734	23,996,446	21,615,335	51,023,557
	l1d_tlb_refill	10,668,882	13,508,554	11,361,701	10,919,225	8,864,051	8,529,882	10,585,729	10,477,041	10,377,924	8,704,652	13,669,502	8,790,558	10,146,694	9,975,270	10,616,415	8,978,112	8,929,447	8,904,460
	l1d_cache_inval	19,669	84,135	10,894,892	8,136	52,009	6,025,806	20,393	104,430	11,063,651	13,320	52,182	8,763,666	25,923	118,112	8,621,267	12,888	66,274	9,604,589
	L1-icache-loads	144,779,504,780	142,521,447,487	143,060,794,345	75,844,433,130	76,173,533,301	74,799,692,311	163,897,647,224	164,325,612,071	162,759,142,799	83,320,560,163	83,221,244,367	81,416,174,050	185,207,090,438	184,579,794,725	182,809,178,245	90,995,840,430	90,708,224,662	89,752,431,209
	l1i_cache_refill	29,232,505	15,307,922	16,558,317	16,983,641	12,577,054	13,060,233	21,044,518	16,427,277	16,445,960	15,396,125	13,122,520	13,261,083	22,902,430	17,684,269	17,739,403	16,379,589	14,133,949	13,563,655
	l1i_tlb_refill	946,007	634,050	639,363	578,649	649,566	425,974	701,903	519,478	657,395	547,117	654,493	433,187	740,205	672,567	676,952	599,884	575,739	530,936
ECB 1351MB	L1-dcache-loads	318,486,042,246	315,749,486,446	314,308,286,565	233,471,865,750	230,385,612,381	228,355,547,437	361,167,173,969	360,323,087,339	361,417,598,153	259,355,322,007	258,257,345,538	258,557,642,266	406,923,736,422	407,334,682,517	406,759,505,999	287,765,069,481	288,342,667,979	286,698,996,441
	l1d_cache_refill	98,547,389	78,453,089	175,339,672	97,519,960	79,978,136	339,807,943	90,219,996	80,317,217	243,007,800	78,151,279	95,215,035	848,560,861	129,387,858	90,400,928	181,171,942	97,020,966	75,226,262	194,339,106
	l1d_cache_rd	216,397,606,105	214,422,299,656	214,206,603,582	158,392,533,575	155,987,270,764	156,097,261,899	248,299,079,947	248,124,296,861	248,136,886,435	176,885,589,738	176,781,144,398	175,351,308,930	280,809,324,955	280,448,430,848	280,223,332,806	197,596,375,755	196,421,556,812	197,381,564,225
	l1d_cache_wr	101,617,540,492	100,141,176,777	99,969,246,333	76,216,530,464	74,283,710,985	74,143,873,539	113,532,618,231	113,517,041,781	113,023,183,345	83,092,299,555	82,929,827,532	82,097,889,035	126,574,800,066	126,122,522,768	125,922,514,310	91,219,401,581	91,097,499,337	90,879,558,806
	l1d_cache_wb	41,573,930	37,552,830	91,822,303	46,642,689	43,845,203	144,363,911	51,433,068	36,664,991	104,922,021	37,818,598	52,241,049	287,762,145	52,173,972	39,919,629	92,148,918	50,304,807	39,019,016	94,595,814
	l1d_tlb_refill	29,535,004	21,044,083	19,729,642	33,443,648	15,251,202	17,837,111	20,933,685	20,140,740	20,478,033	17,243,418	17,537,426	17,106,588	21,113,369	21,275,015	21,527,883	22,945,786	17,066,448	17,689,118
	l1d_cache_inval	84,974	154,876	26,640,521	62,646	79,569	14,327,544	60,264	203,994	21,863,033	13,633	94,431	8,017,897	51,931	222,699	17,284,421	19,663	101,895	21,162,661
	L1-icache-loads	289,168,368,232	286,914,593,696	286,843,332,136	155,220,712,744	151,584,949,343	151,284,118,163	328,332,607,946	328,116,630,633	327,666,578,321	166,799,378,085	167,077,420,729	164,790,475,063	369,714,700,264	370,226,213,926	371,456,517,435	181,526,815,838	181,250,087,206	178,394,455,923
	l1i_cache_refill	70,914,158	30,152,388	30,884,259	58,420,980	24,298,082	24,745,930	43,312,996	32,133,115	32,165,358	30,078,977	25,327,157	25,317,129	44,608,652	34,969,334	34,549,604	32,039,222	27,124,911	26,641,595
	l1i_tlb_refill	2,331,951	1,105,408	986,763	2,063,429	1,299,407	1,113,606	1,083,410	1,223,971	1,028,414	859,339	1,280,590	1,051,687	1,418,475	1,300,230	1,035,972	1,151,257	1,303,065	1,104,273

Table C.9: L2 Cache Benchmarks for ECB - Software Optimizations

	Benchmarks	128 Bits Key			192 Bits Key			256 Bits Key		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	770,818,487	229,985,382	330,570,897	236,399,200	251,731,207	303,949,257	229,955,956	276,963,942	327,751,901
	l2d_cache_refill	38,763,780	24,922,130	44,287,445	23,147,010	27,514,900	31,298,357	18,620,613	26,261,436	33,648,372
	l2d_cache_rd	394,358,340	142,303,665	189,780,049	159,207,243	161,075,991	184,828,144	179,989,353	213,663,654	198,939,946
	l2d_cache_wr	308,991,627	69,094,263	107,099,493	75,457,390	73,715,862	93,476,445	74,369,782	97,141,629	95,340,041
	l2d_cache_wb	7,939,682	7,382,606	7,507,595	6,690,451	7,909,887	9,608,161	5,271,614	7,669,235	10,245,482
	l2d_cache_inval	0	0	0	0	0	0	0	0	0
ECB 676MB	l2d_cache	488,070,272	466,874,767	602,007,940	555,406,466	579,208,898	698,839,330	540,580,710	619,794,889	800,776,301
	l2d_cache_refill	45,301,407	41,596,562	54,478,301	62,569,438	62,702,947	64,766,551	58,284,402	54,270,669	92,718,562
	l2d_cache_rd	354,653,360	348,789,966	449,078,097	331,473,245	374,434,039	432,305,830	359,852,755	396,036,073	486,646,968
	l2d_cache_wr	163,658,787	155,365,394	214,743,377	149,172,842	154,875,906	209,888,596	152,691,796	164,851,071	231,501,785
	l2d_cache_wb	13,575,615	11,961,350	16,660,910	12,641,702	14,814,935	18,882,869	10,592,710	14,910,306	22,400,744
	l2d_cache_inval	0	0	0	0	0	0	0	0	0
ECB 1351MB	l2d_cache	907,777,065	961,942,857	1,327,199,458	1,003,438,564	1,122,061,560	1,280,610,764	1,084,772,112	1,126,343,787	1,438,293,663
	l2d_cache_refill	102,919,255	112,574,194	149,213,475	102,088,337	101,763,708	163,390,017	85,148,610	116,136,271	179,252,598
	l2d_cache_rd	631,443,603	628,055,467	865,942,844	711,639,762	786,255,305	901,191,109	758,741,159	788,907,936	975,996,102
	l2d_cache_wr	254,252,724	274,447,228	434,055,135	296,478,545	346,431,958	403,500,174	356,548,664	316,357,916	452,802,927
	l2d_cache_wb	24,802,524	28,050,567	41,335,925	24,353,722	30,378,915	46,072,659	26,244,228	28,710,500	48,826,229
	l2d_cache_inval	0	0	0	0	0	0	0	0	0

Table C.10: L2 Cache Benchmarks for ECB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
		O2			O3			O2			O3			O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	182,179,896	146,524,969	224,568,570	115,673,444	131,032,766	203,191,506	199,429,009	152,783,312	217,061,046	148,682,556	167,465,033	222,851,125	147,621,731	188,535,662	226,647,758	130,002,316	129,650,346	376,885,467
	l2d_cache_refill	34,047,445	21,662,965	25,830,449	19,259,421	21,185,772	20,259,240	41,316,829	21,684,657	22,163,577	20,744,148	31,181,414	23,944,399	20,688,591	40,364,713	21,387,296	20,418,709	21,300,628	33,229,045
	l2d_cache_rd	114,097,552	127,123,263	133,075,484	77,386,682	87,068,371	172,184,243	97,060,912	140,780,260	177,058,575	95,651,435	88,325,144	145,702,977	109,369,222	92,161,551	159,153,582	117,753,608	125,386,404	198,960,323
	l2d_cache_wr	53,195,530	67,574,127	79,797,405	52,805,541	55,321,805	102,898,111	56,604,725	85,539,176	106,713,777	61,182,657	54,630,074	91,092,682	62,112,689	48,046,758	98,108,509	60,692,477	75,611,685	145,027,143
	l2d_cache_wb	6,979,943	6,604,396	8,254,025	6,335,402	6,625,603	5,937,672	6,213,043	6,868,812	6,911,778	6,848,295	7,242,656	7,116,224	6,120,566	6,758,029	6,241,214	6,519,856	6,421,028	8,011,591
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECB 676MB	l2d_cache	382,765,696	326,862,695	430,249,269	284,042,779	271,903,878	436,220,884	290,649,375	340,051,405	508,978,749	261,852,011	322,663,550	414,623,349	483,071,595	483,071,595	483,071,595	483,071,595	483,071,595	483,071,595
	l2d_cache_refill	66,270,904	59,818,585	45,172,853	44,020,499	43,217,635	43,509,812	42,701,825	44,957,595	46,961,798	42,553,296	63,893,143	41,118,141	59,458,893	59,458,893	59,458,893	59,458,893	59,458,893	59,458,893
	l2d_cache_rd	210,775,401	199,342,298	295,886,557	182,716,747	184,176,531	280,843,979	201,236,033	209,864,602	343,067,045	202,478,927	154,905,276	271,371,253	283,689,828	283,689,828	283,689,828	283,689,828	283,689,828	283,689,828
	l2d_cache_wr	106,106,257	95,203,155	179,006,205	115,517,126	100,565,679	155,325,932	123,694,485	145,815,425	209,668,014	116,487,450	96,362,178	169,640,970	169,087,841	169,087,841	169,087,841	169,087,841	169,087,841	169,087,841
	l2d_cache_wb	14,237,784	13,263,924	15,307,469	14,710,469	13,719,750	14,102,190	13,881,701	14,940,485	15,472,448	14,359,171	12,715,253	12,711,622	12,610,646	12,610,646	12,610,646	12,610,646	12,610,646	12,610,646
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECB 1351MB	l2d_cache	581,386,770	554,057,984	901,972,101	569,203,712	561,872,861	866,903,979	571,685,570	629,679,831	957,889,632	593,908,343	578,485,478	903,982,397	631,272,075	612,189,215	906,804,028	515,788,743	604,775,147	939,107,384
	l2d_cache_refill	88,801,143	102,922,997	103,798,791	97,040,062	103,813,191	81,584,460	83,764,125	103,387,076	90,316,490	103,351,893	104,992,082	103,571,161	102,043,348	88,925,952	92,199,442	80,708,696	105,908,196	93,090,601
	l2d_cache_rd	421,963,936	377,269,523	601,169,241	386,655,649	371,204,654	592,359,298	402,554,498	390,653,773	650,208,787	374,252,063	383,653,939	598,225,917	423,646,398	421,481,637	610,883,528	370,455,393	394,517,701	620,028,637
	l2d_cache_wr	206,410,571	200,201,563	349,307,285	206,325,332	206,343,746	354,412,708	220,517,669	226,125,305	387,083,544	212,878,537	213,043,053	348,917,044	227,202,753	239,728,866	367,039,256	208,993,616	223,621,644	381,603,317
	l2d_cache_wb	29,855,881	25,943,452	25,295,105	28,207,317	26,298,154	26,411,459	27,784,653	26,079,822	28,632,108	28,141,440	26,764,136	26,964,010	25,098,864	28,569,973	27,053,441	27,409,239	27,820,237	26,407,731
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.11: L2 Cache Benchmarks for ECB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**			**192 Bits Key**			**256 Bits Key**		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	184,547,987	158,776,343	261,617,433	209,060,128	174,780,999	258,604,881	183,520,338	177,530,923	264,840,315
	l2d_cache_refill	22,446,669	22,305,402	39,009,912	21,293,701	22,011,331	35,636,750	19,552,420	23,100,034	32,686,335
	l2d_cache_rd	128,426,284	115,052,498	149,654,027	153,851,014	126,825,845	160,077,301	138,622,921	126,302,686	232,119,183
	l2d_cache_wr	71,277,061	66,026,985	80,191,367	78,002,297	69,497,632	81,820,003	58,794,511	51,633,640	125,966,731
	l2d_cache_wb	7,047,976	7,362,289	13,025,210	6,334,120	7,228,028	11,874,154	6,499,577	7,597,516	10,620,081
	l2d_cache_inval	0	0	0	0	0	0	0	0	0
ECB 676MB	l2d_cache	390,508,554	471,558,923	486,955,113	651,726,937	378,920,805	634,376,581	397,297,545	534,311,451	487,967,948
	l2d_cache_refill	49,433,893	56,694,137	75,562,306	71,458,285	43,614,029	106,188,349	47,899,534	76,565,956	74,392,578
	l2d_cache_rd	253,214,438	279,001,936	288,750,061	325,749,221	312,404,534	352,290,612	250,861,438	292,486,685	369,340,810
	l2d_cache_wr	119,746,964	163,388,479	166,386,680	208,518,317	165,061,035	206,979,105	148,583,131	137,671,639	180,447,156
	l2d_cache_wb	15,196,402	16,208,863	19,553,468	20,688,691	13,123,262	31,077,981	15,105,887	16,144,010	25,954,222
	l2d_cache_inval	0	0	0	0	0	0	0	0	0
ECB 1351MB	l2d_cache	2,694,009,953	665,601,478	1,032,557,969	792,355,294	783,992,492	1,031,961,523	855,727,251	658,247,062	3,032,462,419
	l2d_cache_refill	115,196,201	115,300,890	168,867,814	121,129,903	119,942,571	168,376,204	110,044,894	93,698,320	168,972,873
	l2d_cache_rd	1,443,921,639	406,063,352	584,223,073	463,449,390	475,361,990	677,380,424	531,936,903	494,375,888	1,635,599,733
	l2d_cache_wr	1,201,760,613	252,284,897	346,695,527	227,433,298	276,910,544	387,323,438	291,215,157	252,477,829	1,340,532,207
	l2d_cache_wb	33,508,723	31,611,360	53,443,838	32,137,263	33,191,849	61,339,991	28,566,229	34,237,070	52,698,260
	l2d_cache_inval	0	0	0	0	0	0	0	0	0

Table C.12: L2 Cache Benchmarks for ECB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2			O3		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	l2d_cache	105,986,017	124,337,343	152,502,882	152,702,563	152,991,597	279,011,016	132,235,135	92,258,120	158,316,053	121,439,056	97,941,245	201,940,390	106,560,663	114,833,474	191,410,107	123,166,580	151,041,924	152,314,643
	l2d_cache_refill	20,954,789	28,453,840	25,875,818	34,683,802	38,366,155	29,985,778	26,912,688	20,271,113	31,050,970	21,470,889	21,258,958	32,963,184	24,308,405	24,070,917	30,389,660	23,849,214	29,558,828	26,916,875
	l2d_cache_rd	69,660,355	70,388,086	90,977,445	65,806,497	51,955,411	126,736,372	59,811,584	57,175,977	133,734,006	104,836,949	52,453,437	87,155,616	61,124,542	77,157,573	97,164,276	58,854,319	69,697,692	87,730,650
	l2d_cache_wr	37,306,933	37,397,646	61,274,813	34,879,377	35,835,021	105,750,490	37,035,843	34,209,140	77,715,662	64,867,693	28,668,776	60,650,282	36,511,938	44,098,983	65,115,236	37,298,280	62,904,305	65,581,190
	l2d_cache_wb	8,640,019	9,869,844	9,973,644	11,650,435	8,104,002	11,842,856	9,156,680	7,134,520	12,344,808	7,081,145	7,285,052	11,566,957	7,033,314	8,460,739	9,912,486	7,639,295	9,201,985	11,174,711
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECB 676MB	l2d_cache	215,820,127	260,674,203	361,749,744	329,301,159	232,681,915	606,078,093	207,916,444	190,212,795	417,803,898	316,577,833	273,689,469	322,906,280	333,812,628	278,207,874	404,775,981	262,082,791	189,625,367	310,243,507
	l2d_cache_refill	36,428,776	60,244,630	72,932,862	80,114,585	62,134,695	93,991,523	37,468,844	40,815,246	81,995,032	79,082,781	55,746,511	48,285,408	67,135,676	54,629,067	73,682,884	60,250,091	48,363,720	47,677,331
	l2d_cache_rd	199,559,279	117,990,562	198,838,542	115,108,966	101,362,073	273,148,468	126,096,456	116,143,847	210,700,941	109,668,388	156,207,081	233,925,795	159,609,245	177,976,956	253,580,047	119,509,337	105,512,903	218,932,210
	l2d_cache_wr	129,932,081	74,227,461	133,268,080	74,511,494	71,067,208	213,187,113	85,378,895	79,784,744	143,176,327	70,938,404	103,002,533	156,738,645	97,335,359	103,976,972	156,044,138	81,800,139	75,585,031	129,554,945
	l2d_cache_wb	14,207,014	16,695,956	21,779,977	19,637,630	14,763,459	28,123,178	14,082,838	14,412,198	26,028,839	20,130,971	19,001,011	20,415,201	16,129,155	16,903,805	27,942,880	14,367,429	13,389,992	21,552,298
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECB 1351MB	l2d_cache	509,096,413	403,651,389	621,257,610	526,348,083	516,144,699	970,902,191	382,274,662	454,116,243	903,043,323	475,688,995	379,466,423	2,084,561,028	650,443,445	457,451,625	662,453,381	467,428,833	456,245,659	764,191,712
	l2d_cache_refill	103,608,438	95,413,450	100,135,562	114,292,598	132,235,935	104,885,547	78,264,873	112,305,282	161,659,606	116,912,181	80,534,896	165,618,417	131,641,747	111,085,684	135,806,958	99,099,867	112,228,237	129,846,939
	l2d_cache_rd	347,435,999	291,048,581	434,209,440	318,633,135	228,912,744	602,804,141	292,001,392	284,795,896	497,702,714	233,454,717	290,738,788	1,055,450,352	333,937,936	306,181,943	477,537,372	249,293,225	263,430,659	428,301,567
	l2d_cache_wr	186,316,054	165,664,247	280,775,326	176,401,799	151,585,634	459,226,790	178,201,058	165,606,024	322,651,873	141,193,391	197,804,462	942,871,831	220,650,956	180,637,870	287,440,955	166,317,095	173,236,541	295,922,466
	l2d_cache_wb	30,535,702	35,452,628	41,348,633	31,870,248	36,993,101	43,876,272	29,446,970	33,575,393	52,193,809	34,646,519	30,055,371	53,665,995	37,666,754	31,776,107	54,241,027	29,711,102	37,303,963	48,256,829
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.13: Branch Prediction and Speculation Benchmarks for ECB - Software Optimizations

	Benchmarks	**128 Bits Key**			**192 Bits Key**			**256 Bits Key**		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	65,054,116,184	65,457,627,240	64,231,942,793	75,077,932,923	75,241,416,064	73,675,902,706	84,647,124,634	84,936,289,977	84,261,884,103
	branch-misses	296,532,692	346,674,307	233,240,049	360,298,887	392,720,014	249,357,830	441,208,489	431,839,759	270,941,250
	br_immed_spec	57,954,968,932	58,643,735,463	57,257,007,069	67,428,197,694	67,999,890,260	66,110,237,632	76,786,804,040	78,159,426,854	76,523,296,068
	br_indirect_spec	7,182,103,929	7,308,312,386	7,176,326,583	7,644,218,319	7,826,666,083	7,570,671,473	8,146,241,940	8,346,900,757	8,110,222,398
	br_return_spec	5,622,934,087	5,717,895,781	5,617,709,700	5,950,256,883	6,091,736,196	5,908,647,168	6,332,159,526	6,475,238,091	6,307,884,969
ECB 676MB	br_pred	131,075,526,295	131,340,590,300	128,233,312,462	150,683,708,437	150,116,985,867	148,391,734,999	169,528,627,152	170,427,880,260	167,057,810,402
	branch-misses	725,023,109	710,138,029	470,925,075	840,906,636	763,980,444	503,985,751	869,716,445	952,929,704	534,407,868
	br_immed_spec	116,703,577,410	117,699,224,638	115,063,483,835	135,380,927,021	136,966,552,198	131,813,625,447	153,523,469,490	155,343,536,258	151,589,911,814
	br_indirect_spec	14,318,076,667	14,660,136,038	14,412,206,327	15,352,703,326	15,622,626,594	15,208,352,123	16,240,689,608	16,637,809,564	16,213,840,442
	br_return_spec	11,204,518,584	11,450,517,528	11,306,164,733	11,930,146,806	12,160,553,064	11,877,550,972	12,621,607,119	12,904,029,829	12,613,692,508
ECB 1351MB	br_pred	262,227,264,987	264,089,481,575	257,091,300,970	300,466,217,075	300,109,825,967	294,890,424,259	339,834,556,573	342,233,354,805	332,742,702,505
	branch-misses	1,431,496,142	1,403,447,624	940,672,202	1,476,700,846	1,551,972,407	1,022,826,497	1,765,019,688	1,705,783,604	1,067,819,011
	br_immed_spec	233,430,053,966	234,147,972,973	227,133,774,023	270,086,019,319	273,729,830,455	266,176,534,539	307,657,524,091	309,974,313,776	300,306,722,333
	br_indirect_spec	28,721,503,290	29,151,118,345	28,520,049,771	30,605,606,207	31,266,598,939	30,503,154,577	32,528,735,446	33,230,923,760	32,250,911,263
	br_return_spec	22,421,615,390	22,784,402,483	22,358,950,927	23,876,049,712	24,333,564,386	23,808,686,921	25,229,951,521	25,775,987,702	25,108,680,439

Table C.14: Branch Prediction and Speculation Benchmarks for ECB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2				O3	
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	38,200,751,343	37,606,200,025	37,521,198,909	14,788,589,851	14,639,417,905	14,921,471,205	43,439,557,934	42,743,437,802	43,295,099,198	15,442,871,934	15,349,961,267	16,003,160,406	48,831,557,909	48,242,550,068	48,412,182,212	16,226,796,596	16,151,436,058	16,224,674,379
	branch-misses	212,252,911	250,425,781	245,471,278	132,174,945	131,788,212	133,110,726	237,826,106	237,541,535	278,754,847	133,261,723	134,151,287	137,084,134	333,937,246	265,400,423	253,962,835	137,094,726	134,136,330	133,186,142
	br_immed_spec	36,205,551,185	36,040,827,632	36,472,335,059	13,121,837,611	13,070,840,057	13,329,092,258	41,588,724,495	41,463,579,165	41,315,853,948	13,781,352,207	13,687,234,186	13,681,435,494	47,154,425,825	46,521,743,435	47,224,055,162	14,400,672,088	14,179,758,287	14,103,659,296
	br_indirect_spec	1,708,646,290	1,705,286,121	1,752,209,120	1,849,041,623	1,834,401,183	1,888,172,664	1,752,568,635	1,728,093,081	1,789,669,131	1,919,097,158	1,898,769,104	1,936,837,901	1,772,669,791	1,745,784,013	1,835,821,822	1,983,831,824	1,966,730,304	1,973,691,337
	br_return_spec	1,090,360,311	1,072,464,430	1,125,924,430	1,221,186,578	1,210,720,722	1,250,574,383	1,120,140,898	1,098,844,594	1,152,957,178	1,286,449,724	1,271,771,561	1,304,846,883	1,139,034,627	1,124,795,281	1,194,218,826	1,348,334,788	1,334,491,759	1,353,260,759
ECB 676MB	br_pred	76,428,670,474	76,143,192,137	75,381,494,723	29,842,318,478	75,599,784,704	30,072,617,164	86,593,214,719	86,050,919,641	86,269,633,941	31,189,046,809	31,051,099,734	32,079,084,053	97,575,459,124	97,141,523,160	96,976,882,502	32,510,695,375	32,175,364,359	32,779,863,015
	branch-misses	439,821,338	484,255,378	462,758,324	272,854,956	468,149,825	272,929,085	605,717,887	479,138,677	502,182,043	272,018,600	268,109,701	276,474,187	573,713,816	529,376,287	514,372,248	270,643,490	269,660,276	267,991,552
	br_immed_spec	72,561,811,645	72,194,700,765	72,774,105,828	26,386,853,780	72,158,350,149	26,873,801,734	83,292,536,128	83,119,886,015	83,552,792,109	27,395,449,147	27,370,844,764	27,264,489,069	94,262,551,229	94,058,884,011	94,012,641,760	28,624,718,870	28,495,693,500	28,621,861,927
	br_indirect_spec	3,442,806,034	3,396,881,812	3,509,863,665	3,715,685,784	3,402,268,359	3,850,801,823	3,462,187,818	3,462,724,818	3,612,128,927	3,812,890,167	3,792,140,986	3,847,684,714	3,557,690,233	3,504,886,084	3,642,652,575	3,942,592,616	3,929,506,904	4,017,709,731
	br_return_spec	2,186,648,567	2,156,369,190	2,247,326,795	2,461,751,540	2,145,239,095	2,563,747,029	2,210,598,963	2,213,986,036	2,330,215,307	2,562,287,066	2,541,245,496	2,607,109,291	2,287,127,343	2,265,204,481	2,382,411,585	2,681,197,166	2,673,054,517	2,763,961,732
ECB 1351MB	br_pred	152,513,889,113	151,553,968,112	152,908,724,554	60,667,690,461	151,458,862,798	60,869,989,550	173,294,525,925	172,950,125,794	174,358,947,994	62,226,820,312	62,345,228,611	63,959,416,069	194,887,899,291	194,595,280,783	195,725,070,913	64,824,961,229	64,402,279,872	65,996,676,167
	branch-misses	916,835,045	867,206,686	855,517,067	547,815,359	895,352,258	543,561,463	966,701,610	1,021,165,892	918,252,723	541,528,170	541,223,162	549,808,574	1,281,607,880	975,587,181	1,120,017,772	542,227,771	532,631,914	545,420,854
	br_immed_spec	145,449,153,228	144,609,239,124	145,568,876,619	53,449,947,724	144,763,543,141	54,034,829,117	166,620,229,632	166,172,003,414	167,675,242,424	54,886,544,278	54,464,863,683	55,607,773,954	188,303,465,594	187,760,854,928	188,592,653,055	57,189,776,213	56,700,542,314	58,444,030,140
	br_indirect_spec	6,909,099,169	6,731,655,683	7,087,681,399	7,590,707,369	6,809,129,749	7,738,494,877	6,927,492,629	6,920,848,746	7,272,231,588	7,645,227,615	7,568,515,419	7,858,470,640	7,080,329,979	7,054,571,351	7,338,060,948	7,890,801,997	7,831,205,727	8,177,562,794
	br_return_spec	4,389,610,456	4,252,027,982	4,552,309,334	5,043,832,429	4,302,296,655	5,159,970,780	4,426,790,169	4,408,402,656	4,708,839,046	5,135,838,347	5,096,305,748	5,329,566,476	4,564,124,305	4,543,383,958	4,782,972,018	5,380,174,629	5,347,714,318	5,602,737,402

Table C.15: Branch Prediction and Speculation Benchmarks for ECB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**			**192 Bits Key**			**256 Bits Key**		
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	65,640,389,063	65,647,040,679	64,627,463,478	72,265,927,033	75,317,400,255	75,923,059,549	84,666,814,422	85,293,075,241	85,412,236,062
	branch-misses	385,922,270	349,793,669	310,035,867	348,247,018	382,085,057	359,837,364	429,254,884	430,397,280	363,230,494
	br_immed_spec	58,605,391,552	58,421,954,109	58,226,716,495	64,841,617,502	67,757,268,447	66,892,795,047	76,725,988,906	77,030,579,954	75,136,950,440
	br_indirect_spec	7,212,096,549	7,236,757,476	7,129,885,336	7,419,693,524	7,732,690,309	7,614,818,835	8,121,412,976	8,227,583,997	8,065,470,140
	br_return_spec	5,624,819,229	5,653,841,742	5,583,683,247	5,782,209,339	6,006,547,268	5,944,966,756	6,300,863,919	6,381,566,687	6,280,496,183
ECB 676MB	br_pred	131,329,849,766	132,676,015,622	131,208,170,481	150,215,161,073	151,691,454,428	150,818,351,393	169,963,878,390	171,718,422,341	170,334,091,737
	branch-misses	772,339,859	702,701,149	687,340,051	781,433,773	773,002,719	783,386,008	918,799,534	860,439,554	853,722,936
	br_immed_spec	117,063,345,449	117,413,765,300	116,690,510,611	134,937,238,900	134,229,162,899	135,131,909,696	153,718,726,502	152,917,287,087	153,807,629,501
	br_indirect_spec	14,318,264,516	14,498,772,332	14,413,267,400	15,182,704,223	15,388,456,720	15,303,384,934	16,113,587,185	16,374,360,424	16,341,205,716
	br_return_spec	11,180,435,420	11,340,014,411	11,261,456,257	11,817,088,530	11,970,999,270	11,897,333,626	12,513,090,547	12,700,041,666	12,656,284,939
ECB 1351MB	br_pred	259,545,675,638	262,711,983,053	262,181,435,301	299,174,803,166	301,719,545,137	301,298,066,421	338,481,203,919	340,871,949,020	341,480,969,433
	branch-misses	1,208,196,917	1,407,028,791	1,381,845,954	1,387,925,952	1,538,755,276	1,543,419,634	1,642,148,859	1,707,800,918	1,710,872,509
	br_immed_spec	231,110,951,730	233,880,664,249	233,595,527,561	268,943,901,601	270,874,152,757	270,970,854,920	306,339,136,877	308,336,278,078	307,942,997,944
	br_indirect_spec	28,559,026,724	28,929,805,194	28,876,913,414	30,252,187,923	30,889,204,527	30,838,857,205	32,364,194,236	32,896,791,868	32,719,551,368
	br_return_spec	22,335,567,049	22,603,018,740	22,561,017,556	23,589,893,125	24,026,486,810	23,975,114,481	25,072,214,065	25,498,276,612	25,356,982,240

Table C.16: Branch Prediction and Speculation Benchmarks for ECB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key						192 Bits Key						256 Bits Key					
			O2			O3			O2			O3			O2				O3	
		Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads	Normal	OpenMP	Threads
ECB 115MB	br_pred	37,419,388,597	37,256,976,111	36,635,271,535	14,600,726,794	14,211,422,149	13,776,039,500	43,032,659,709	42,370,619,157	42,187,042,984	15,221,613,119	14,865,374,619	14,715,290,339	48,345,787,488	48,186,371,101	47,642,529,687	15,781,358,151	15,731,396,912	15,096,099,851
	branch-misses	204,690,333	254,457,795	212,653,989	132,027,589	130,746,650	126,028,485	235,856,458	249,619,056	266,360,135	133,932,828	131,946,507	127,657,891	244,542,276	250,822,820	261,315,538	133,178,712	130,595,116	127,539,054
	br_immed_spec	36,203,828,139	35,282,231,055	35,229,583,673	12,801,035,068	12,812,340,797	12,354,240,301	41,286,871,790	40,992,855,961	40,541,764,696	13,411,801,615	13,185,942,603	12,883,157,294	46,508,322,851	46,196,410,542	46,269,947,985	13,940,357,202	13,754,426,613	13,653,026,782
	br_indirect_spec	1,658,035,461	1,571,676,490	1,620,194,399	1,774,511,575	1,795,667,185	1,720,656,839	1,657,667,146	1,661,318,477	1,669,446,442	1,842,313,816	1,807,521,622	1,780,178,816	1,695,514,925	1,643,602,798	1,690,425,697	1,901,349,055	1,860,917,134	1,862,097,383
	br_return_spec	1,058,358,642	984,677,342	1,022,946,725	1,170,963,715	1,170,475,117	1,143,236,539	1,052,100,916	1,057,198,900	1,065,815,333	1,233,991,261	1,211,158,461	1,194,133,644	1,088,532,754	1,051,184,929	1,089,952,877	1,292,905,804	1,262,413,117	1,273,146,656
ECB 676MB	br_pred	75,456,233,149	74,596,854,405	74,213,533,131	29,189,814,029	28,787,823,004	28,853,261,029	85,567,597,244	85,364,104,996	84,727,339,310	30,294,406,398	30,065,911,110	29,739,944,330	96,820,705,894	96,273,509,650	95,803,893,030	31,450,218,101	31,265,710,406	31,109,878,948
	branch-misses	431,826,925	480,493,897	434,702,226	268,038,779	263,656,134	263,380,493	433,487,808	506,028,538	479,829,308	265,765,069	263,612,576	262,135,717	466,389,690	514,482,701	551,363,685	264,483,018	264,796,269	261,684,979
	br_immed_spec	72,334,892,022	71,717,391,332	70,971,762,460	25,493,970,113	25,517,029,957	24,759,090,171	82,843,896,174	82,282,898,629	82,265,369,034	26,828,490,314	26,946,085,088	26,792,356,548	93,079,270,312	93,229,038,357	92,755,154,103	27,954,019,450	27,926,472,430	27,984,402,563
	br_indirect_spec	3,334,241,256	3,296,898,662	3,252,871,079	3,543,687,265	3,565,065,097	3,427,910,696	3,371,647,704	3,341,259,663	3,322,157,043	3,690,047,032	3,703,647,410	3,692,455,875	3,358,778,605	3,401,076,794	3,421,163,594	3,815,832,142	3,816,140,939	3,833,412,527
	br_return_spec	2,099,775,261	2,059,913,712	2,052,452,402	2,339,852,171	2,339,540,817	2,266,426,131	2,134,518,808	2,115,012,305	2,121,134,430	2,468,943,295	2,465,428,628	2,468,792,931	2,142,892,410	2,178,148,813	2,200,718,325	2,592,635,315	2,588,104,301	2,600,395,012
ECB 1351MB	br_pred	151,264,078,847	149,959,331,231	149,516,378,572	59,261,539,206	58,107,636,701	58,067,778,098	171,529,198,611	171,249,204,004	171,685,461,987	60,649,845,966	60,272,650,385	60,090,498,836	193,459,069,991	193,244,255,195	192,851,509,459	63,084,031,918	63,049,448,427	63,235,085,389
	branch-misses	823,761,605	841,712,734	892,197,462	537,156,937	530,467,338	529,521,953	887,637,932	971,245,723	940,896,387	531,387,917	525,733,207	526,525,959	944,250,747	947,617,392	1,058,643,722	531,992,908	530,451,043	533,167,994
	br_immed_spec	144,775,805,826	143,248,702,809	142,697,242,576	52,240,183,210	50,965,562,883	50,694,924,237	165,692,532,105	165,178,079,502	164,473,706,959	53,765,012,845	53,449,606,062	53,438,568,691	186,677,197,824	186,298,038,483	186,110,564,564	56,028,105,899	55,739,016,787	54,882,295,151
	br_indirect_spec	6,726,473,955	6,523,882,025	6,567,547,426	7,365,532,145	7,076,077,321	7,063,756,869	6,715,196,883	6,672,972,050	6,655,420,997	7,399,230,812	7,389,372,637	7,351,736,907	6,746,050,727	6,771,317,407	6,843,391,471	7,651,862,621	7,587,210,775	7,484,352,097
	br_return_spec	4,243,898,422	4,071,364,378	4,130,979,662	4,881,375,037	4,654,640,037	4,633,426,311	4,249,562,740	4,223,763,519	4,235,296,654	4,947,186,254	4,933,019,536	4,917,347,273	4,324,748,397	4,327,999,647	4,395,037,779	5,194,765,354	5,159,241,671	5,094,806,520

Appendix D

Tables - CBC

Table D.1: General Benchmarks for CBC - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	context-switches	21,763	15,907	25,613	19,087	31,277	22,412
	page-faults	121,178	121,195	121,178	121,193	121,179	121,194
	cycles	452,657,633,951	456,460,457,533	529,111,205,349	536,574,366,489	612,026,044,997	617,400,220,684
	instructions	825,697,169,534	825,368,232,001	978,316,700,678	975,440,566,897	1,128,032,282,019	1,124,712,080,258
	time elapsed (s)	253.410614719	173.024025781	296.232265590	202.409491683	342.750457546	235.564739316
	cache-misses	43,601,479	43,895,862	56,008,476	46,527,026	57,715,126	52,318,322
	mem_access	352,495,981,502	349,343,457,127	413,870,666,398	410,819,548,898	477,375,764,510	473,839,045,682
	mem_access_rd	278,074,248,205	275,702,438,803	329,041,869,045	325,952,508,032	380,970,925,141	375,459,418,842
	mem_access_wr	74,081,183,182	73,491,026,319	85,165,189,238	84,253,535,197	96,470,623,163	94,867,261,698
CBC 676 MB	context-switches	42,922	34,667	51,196	42,043	62,931	47,466
	page-faults	242,234	242,249	242,235	242,251	242,234	242,250
	cycles	902,108,769,086	919,755,772,047	1,068,477,955,210	1,080,514,066,863	1,221,918,897,527	1,239,369,910,190
	instructions	1,652,654,882,325	1,657,579,070,441	1,956,828,259,587	1,958,428,697,362	2,258,811,578,054	2,257,912,804,662
	time elapsed (s)	505.010304178	361.558635826	598.121148646	427.564794255	684.301936803	492.929702485
	cache-misses	104,704,837	106,684,448	110,049,929	107,745,286	105,555,258	125,872,745
	mem_access	704,040,818,057	700,960,551,103	829,922,371,009	826,820,578,099	955,498,137,784	945,368,237,055
	mem_access_rd	555,973,955,674	556,728,252,604	659,185,111,878	656,663,374,136	762,123,831,531	752,466,650,967
	mem_access_wr	148,311,857,594	148,335,461,233	170,776,757,425	169,873,135,274	193,042,146,590	190,601,299,368
CBC 1135 MB	context-switches	86,739	72,360	98,559	81,920	116,760	95,437
	page-faults	485,261	485,365	484,344	484,466	484,342	484,357
	cycles	1,805,931,101,781	1,831,305,667,874	2,123,640,844,462	2,158,304,975,573	2,438,249,995,244	2,481,076,803,002
	instructions	3,305,959,699,612	3,308,759,075,314	3,911,869,589,280	3,917,102,558,135	4,517,272,037,445	4,520,267,919,658
	time elapsed (s)	1011.132696467	746.607332710	1190.686367616	877.695822272	1364.945040583	1008.094445788
	cache-misses	180,517,807	206,629,761	207,208,092	245,013,531	260,835,473	270,149,853
	mem_access	1,408,329,166,075	1,395,212,542,989	1,657,443,028,137	1,646,584,604,434	1,908,243,301,775	1,898,887,884,881
	mem_access_rd	1,111,494,171,196	1,104,582,403,865	1,316,903,973,827	1,303,255,455,456	1,522,966,590,302	1,522,952,141,981
	mem_access_wr	296,557,366,186	294,957,407,862	340,525,860,466	337,701,299,526	385,555,233,361	385,005,803,856

Table D.2: General Benchmarks for CBC - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	context-switches	7,232	6,057	3,154	2,793	5,943	4,785	3,543	3,075	7,773	5,688	3,802	3,279
	page-faults	121,178	121,192	121,178	121,193	121,178	121,193	121,178	121,193	121,178	121,195	121,178	121,195
	cycles	112,330,170,196	114,373,325,531	66,749,227,368	66,224,993,279	126,245,894,960	126,084,784,947	74,110,513,543	73,700,687,525	143,045,816,295	142,352,183,295	81,205,348,029	79,726,200,239
	instructions	217,460,158,808	216,013,530,195	122,090,379,666	116,425,714,857	249,783,067,874	247,974,864,005	135,697,924,735	131,794,299,320	282,525,413,219	282,430,013,560	148,576,922,144	144,169,258,536
	time elapsed (s)	63.436989077	49.428489224	37.408879056	31.043504570	70.709116381	52.632080767	41.553918029	33.558967796	80.192315606	57.981089754	45.504241070	35.920834158
	cache-misses	159,175,236	388,423,628	25,805,205	29,908,461	27,237,810	27,155,456	25,295,300	24,300,048	28,471,686	25,964,676	27,597,148	22,289,101
	mem_access	80,335,872,291	80,098,711,432	58,545,411,751	57,993,087,605	92,087,342,033	91,555,594,798	66,153,632,194	65,616,372,669	103,120,345,648	102,605,598,435	73,196,447,911	71,707,433,957
	mem_access_rd	55,115,818,145	54,660,377,195	39,449,268,881	38,254,002,634	62,566,604,157	62,121,796,695	44,719,482,652	43,600,042,747	70,954,990,021	70,857,748,853	49,878,956,730	49,555,559,039
	mem_access_wr	25,995,010,112	25,790,768,038	19,161,736,416	18,617,800,319	28,774,466,118	28,599,480,036	21,255,634,020	20,889,975,690	32,121,189,197	31,999,711,575	23,143,306,646	23,203,486,458
CBC 676 MB	context-switches	14,685	12,347	6,478	5,539	12,101	9,558	9,558	6,057	13,646	10,725	7,787	6,550
	page-faults	242,234	242,249	242,234	242,250	242,234	242,249	242,249	242,249	242,235	242,249	242,234	242,250
	cycles	223,438,018,980	222,105,527,392	133,797,182,398	133,885,379,555	254,052,956,746	251,966,862,391	251,966,862,391	149,291,898,978	286,576,693,392	284,501,843,323	164,499,096,004	161,648,426,243
	instructions	436,403,509,738	435,323,267,979	242,865,454,325	237,014,031,386	498,479,296,999	498,137,564,194	498,137,564,194	266,392,764,954	563,344,866,710	563,227,766,315	296,774,749,984	291,706,605,992
	time elapsed (s)	126.170184770	94.537819957	74.985033373	62.201525668	142.265783679	104.176433583	104.176433583	67.082273741	160.502138431	116.055716029	92.147456406	71.786979360
	cache-misses	64,280,399	86,808,285	48,547,839	50,183,964	58,347,871	58,195,560	58,195,560	53,372,124	72,820,288	61,221,140	47,967,907	56,783,831
	mem_access	161,068,206,366	160,017,003,390	117,500,022,693	116,279,018,259	183,854,283,672	183,346,481,298	183,346,481,298	130,460,415,775	206,762,751,919	206,002,612,676	145,977,280,498	145,309,456,277
	mem_access_rd	110,000,812,809	109,794,562,690	79,410,913,107	78,577,298,405	125,455,536,625	125,148,792,356	125,148,792,356	89,073,659,454	141,701,839,819	141,528,897,121	99,602,975,870	98,449,373,891
	mem_access_wr	51,842,343,924	51,735,079,781	38,331,994,936	38,126,350,967	57,774,787,915	57,664,633,035	57,664,633,035	42,063,073,530	64,161,094,355	64,114,349,789	46,412,491,162	46,117,408,565
CBC 1135 MB	context-switches	28,782	24,809	13,352	10,910	25,431	19,232	14,054	11,830	28,728	21,434	15,928	13,189
	page-faults	484,344	484,357	484,342	484,356	484,342	484,359	484,342	484,359	484,342	484,357	484,342	484,356
	cycles	445,563,967,892	443,213,289,189	269,897,516,236	266,074,249,709	507,592,058,756	503,931,288,883	295,719,352,311	294,791,784,831	570,551,943,300	568,748,981,293	328,931,741,261	328,963,176,404
	instructions	872,272,960,411	869,097,340,221	485,726,917,007	478,518,277,011	997,721,281,351	994,581,842,608	541,231,968,615	531,113,944,666	1,126,875,808,077	1,126,569,332,283	594,252,950,561	582,797,874,891
	time elapsed (s)	250.980778547	188.300606472	151.232297471	122.251178924	284.318503964	208.322872624	165.645262801	132.646371822	319.540771744	231.613808976	184.246355649	146.385042670
	cache-misses	112,312,837	114,097,707	128,239,161	107,412,952	136,021,568	120,179,890	101,829,463	97,899,460	122,309,871	119,397,656	94,305,337	1,064,809,314
	mem_access	323,146,839,169	321,820,371,801	235,607,293,291	233,025,429,155	367,524,982,080	367,335,605,656	263,858,946,960	261,779,157,473	413,069,908,168	412,329,373,528	292,077,501,708	290,701,077,948
	mem_access_rd	219,698,971,564	219,475,700,932	158,387,198,098	157,948,878,046	251,425,978,197	250,170,739,164	178,809,248,047	178,089,170,164	283,842,790,622	283,668,966,230	199,410,652,757	198,475,519,434
	mem_access_wr	103,639,525,996	103,499,003,175	76,593,543,961	76,370,407,086	115,824,952,196	115,185,873,645	84,868,356,759	84,641,717,561	128,577,814,748	128,582,292,040	93,018,563,227	92,222,408,818

Table D.3: General Benchmarks for CBC with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	context-switches	21,496	16,382	25,388	19,308	29,199	22,181
	page-faults	29,702	4,787	6,273	4,852	5,268	5,458
	cycles	455,732,401,052	450,780,849,765	533,093,555,430	532,569,228,901	611,524,993,527	608,291,428,128
	instructions	826,240,801,444	798,454,353,304	975,877,084,329	950,737,948,081	1,127,036,735,743	1,088,715,036,213
	time elapsed (s)	255.118148131	173.092235629	298.410256390	203.641575476	342.284537544	233.833099263
	cache-misses	50,453,636	48,213,324	43,003,158	37,191,432	44,920,291	41,744,868
	mem_access	352,998,034,087	343,263,136,403	415,646,771,534	404,770,775,968	477,686,303,249	460,903,614,983
	mem_access_rd	278,769,786,922	271,796,167,647	330,499,773,303	321,536,761,644	380,944,833,096	370,925,552,194
	mem_access_wr	74,209,788,886	70,915,326,125	85,395,615,469	81,346,749,547	96,282,384,503	91,661,303,478
CBC 676 MB	context-switches	43,413	35,353	50,834	40,048	61,766	46,105
	page-faults	32,216	5,705	7,129	5,818	5,334	5,833
	cycles	909,221,057,907	903,091,682,540	1,064,823,889,458	1,058,645,699,865	1,230,235,341,667	1,219,907,573,987
	instructions	1,647,996,120,743	1,597,401,618,099	1,950,154,925,620	1,887,086,803,726	2,255,580,868,118	2,184,755,131,972
	time elapsed (s)	508.958926340	356.860918281	596.061660742	420.669437442	688.795360467	482.351696789
	cache-misses	87,253,338	76,430,360	107,269,878	88,927,355	121,389,452	125,033,884
	mem_access	706,864,124,820	693,695,012,052	830,623,737,338	812,821,780,388	956,742,685,477	929,425,875,904
	mem_access_rd	557,974,868,308	547,342,072,321	660,194,943,996	648,014,269,593	763,646,629,463	744,599,695,409
	mem_access_wr	148,690,921,331	142,929,930,537	170,607,122,400	163,507,862,446	193,209,332,369	183,807,783,115
CBC 1135 MB	context-switches	84,825	68,062	108,369	80,869	116,632	92,494
	page-faults	52,321	7,188	6,229	7,989	6,639	7,359
	cycles	1,811,789,176,362	1,812,522,331,956	2,136,119,167,703	2,127,626,285,261	2,442,177,327,032	2,443,573,140,531
	instructions	3,300,062,078,721	3,210,575,923,319	3,903,677,546,350	3,796,013,880,456	4,512,358,300,506	4,381,240,541,282
	time elapsed (s)	1014.104079007	712.340239593	1196.064487008	832.717552941	1367.045220805	951.026820896
	cache-misses	198,054,087	166,136,085	191,924,822	202,980,807	205,262,404	168,578,771
	mem_access	1,411,284,783,575	1,377,368,379,969	1,661,433,646,749	1,617,331,772,175	1,910,424,752,679	1,853,644,604,330
	mem_access_rd	1,114,118,471,614	1,090,719,734,499	1,320,290,291,071	1,287,624,038,634	1,525,657,756,506	1,487,711,507,866
	mem_access_wr	296,667,961,466	284,854,564,216	341,628,482,359	325,971,145,169	386,009,510,940	367,886,516,411

Table D.4: General Benchmarks for CBC with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	context-switches	7,240	4,243	3,347	2,726	6,722	5,288	3,514	2,975	6,735	5,253	4,129	3,412
	page-faults	5,740	5,572	4,851	4,771	4,836	4,868	5,556	4,851	4,740	4,898	5,475	5,523
	cycles	110,229,544,377	109,080,044,863	66,297,121,496	66,084,834,323	125,295,193,795	123,498,844,754	73,417,984,283	72,369,858,250	141,979,139,276	140,258,599,986	80,092,631,885	79,438,131,852
	instructions	215,717,286,313	212,921,208,403	121,212,054,449	118,359,630,866	248,302,728,840	245,472,962,765	133,861,628,746	131,020,937,545	280,126,546,413	277,311,373,350	147,101,704,738	142,604,299,662
	time elapsed (s)	62.324189685	46.208793642	37.165683350	30.365328102	70.215397226	51.567612336	41.160414337	32.668458794	79.507528324	57.340362626	44.882906975	35.509688611
	cache-misses	38,714,145	20,907,882	206,968,309	15,436,921	24,522,415	31,500,735	26,376,136	18,939,546	21,583,484	28,023,457	19,278,931	70,422,650
	mem_access	80,178,407,381	79,218,213,931	58,251,789,699	55,934,658,169	91,399,128,388	90,786,554,363	65,402,546,377	64,500,782,658	102,824,409,435	102,772,105,238	72,712,240,277	71,406,786,006
	mem_access_rd	54,702,970,071	54,088,193,928	39,060,061,738	39,129,129,123	62,781,178,603	62,744,484,374	44,584,059,863	44,150,507,005	70,873,653,507	70,005,628,545	49,851,228,666	48,697,870,244
	mem_access_wr	25,648,405,170	25,302,606,113	18,752,692,279	18,844,680,242	28,771,511,174	28,754,975,059	20,925,995,157	20,787,910,662	31,941,054,796	31,513,705,924	23,052,605,917	22,818,044,143
CBC 676 MB	context-switches	14,291	8,460	6,383	5,722	12,177	9,575	7,093	5,938	13,502	10,934	7,103	6,539
	page-faults	8,430	6,367	11,296	5,707	6,171	5,882	6,098	5,897	6,186	6,201	5,690	6,121
	cycles	219,784,412,235	217,512,446,948	131,505,431,162	132,147,138,822	250,126,288,538	249,120,838,779	145,537,781,117	144,949,976,158	282,751,900,247	281,673,430,004	159,990,947,843	161,860,105,845
	instructions	431,854,910,281	430,533,358,095	240,481,464,759	236,343,313,699	495,946,933,684	492,676,883,667	266,143,284,567	260,660,253,679	561,695,439,304	557,754,322,067	292,481,205,703	289,878,577,401
	time elapsed (s)	123.747516418	91.494814982	73.672802774	60.733652527	140.074094391	103.084199485	81.556709658	65.260203268	158.349935690	114.753282040	89.630286224	71.809373018
	cache-misses	62,709,712	72,587,217	46,836,593	42,664,023	54,115,542	46,346,723	49,669,723	40,076,823	77,608,783	57,516,834	39,627,519	43,812,368
	mem_access	160,912,671,364	159,031,541,815	116,378,084,293	114,260,838,455	182,425,343,357	182,431,620,370	130,930,670,188	130,297,061,629	204,780,978,775	204,843,509,404	145,442,724,318	143,158,716,661
	mem_access_rd	108,964,521,445	108,601,724,774	78,839,848,139	78,256,860,720	125,588,577,811	124,498,517,266	89,094,740,547	88,139,608,538	141,917,960,187	141,399,602,373	98,816,211,416	98,933,555,186
	mem_access_wr	51,047,956,310	50,788,728,550	37,962,039,820	37,605,796,743	57,472,988,037	56,992,563,968	41,973,389,747	41,220,143,201	64,013,188,100	63,623,442,032	45,728,406,878	45,404,441,077
CBC 1135 MB	context-switches	28,912	24,408	24,514	10,849	24,300	19,493	13,892	11,901	27,522	21,100	15,556	12,758
	page-faults	110,779	6,966	148,609	6,743	14,749	6,727	14,292	7,653	6,777	7,015	7,622	6,887
	cycles	441,490,206,447	439,399,292,097	267,122,306,789	264,170,664,725	500,754,569,360	499,273,086,172	291,669,939,228	293,264,038,671	564,378,544,911	564,425,970,508	319,874,023,909	319,916,565,684
	instructions	865,688,645,566	860,576,892,619	486,336,701,733	473,208,350,397	992,509,352,812	988,173,069,252	533,116,972,267	524,232,271,244	1,119,823,524,787	1,119,294,866,498	586,629,713,770	576,195,585,471
	time elapsed (s)	248.194673441	186.949445296	154.699955521	121.480121632	280.646840446	205.997108092	163.372968264	131.746552846	316.034374176	230.092800505	179.166882431	141.123138440
	cache-misses	108,073,117	81,689,581	89,372,348	73,928,267	133,330,995	119,019,474	93,045,677	61,505,598	116,745,052	130,209,243	72,168,063	65,674,918
	mem_access	321,481,987,180	320,021,907,954	236,270,631,024	231,507,683,394	364,430,465,602	364,591,746,428	261,892,693,866	259,753,070,651	410,646,093,953	409,832,981,725	289,834,952,276	288,915,797,223
	mem_access_rd	218,533,829,020	217,947,234,229	158,132,973,384	156,202,190,553	250,673,661,755	249,434,489,255	177,776,306,210	176,978,556,895	283,112,314,683	282,373,911,215	198,050,308,789	197,117,590,966
	mem_access_wr	102,606,449,285	102,065,424,524	76,536,040,637	75,032,289,732	114,777,978,850	114,182,106,876	83,648,323,245	82,744,565,111	127,648,172,067	127,296,879,532	91,776,954,719	90,889,907,043

Table D.5: L1 Cache Benchmarks for CBC - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	L1-dcache-loads	352,404,970,227	349,690,799,338	414,064,162,197	411,098,501,041	477,376,985,743	473,874,723,781
	l1d_cache_refill	46,453,477	41,170,739	50,642,118	48,094,864	57,610,786	52,483,943
	l1d_cache_rd	278,187,651,827	275,402,707,783	328,964,036,552	326,933,331,102	380,847,133,178	379,123,537,440
	l1d_cache_wr	74,159,100,261	73,430,601,715	85,158,841,569	84,499,123,124	96,461,340,963	95,557,884,650
	l1d_cache_wb	15,812,398	14,673,458	15,746,592	15,275,966	16,845,909	17,385,355
	l1d_tlb_refill	18,104,520	32,871,869	19,565,394	34,788,360	21,998,787	37,276,584
	l1d_cache_inval	42,668	153,400	37,136	223,441	83,781	264,770
	L1-icache-loads	270,488,869,810	256,018,375,985	316,379,766,390	299,870,980,636	365,530,343,310	346,153,752,719
	l1i_cache_refill	63,395,793	58,264,007	71,115,783	63,006,546	78,300,207	68,827,146
	l1i_tlb_refill	2,204,900	1,701,941	4,649,514	1,530,166	2,311,842	1,865,540
CBC 676MB	L1-dcache-loads	703,949,676,276	699,135,671,337	830,252,913,889	823,381,402,786	955,484,444,995	952,453,389,085
	l1d_cache_refill	107,654,471	108,940,600	109,487,340	104,242,451	103,891,228	128,904,593
	l1d_cache_rd	555,663,499,938	550,336,745,015	658,677,137,481	651,910,655,371	762,415,951,037	756,451,994,194
	l1d_cache_wr	148,299,780,923	147,108,286,255	170,859,232,098	169,483,190,435	193,049,632,639	191,239,266,296
	l1d_cache_wb	33,298,472	34,080,369	29,636,139	34,484,399	31,347,098	36,394,351
	l1d_tlb_refill	40,214,915	73,322,894	40,245,617	72,237,710	45,133,072	86,296,731
	l1d_cache_inval	88,105	416,226	94,229	500,839	168,232	602,740
	L1-icache-loads	538,573,692,016	514,838,481,384	638,203,542,180	604,314,769,723	733,709,541,446	693,601,808,202
	l1i_cache_refill	126,953,024	123,635,343	141,023,781	138,348,532	160,103,704	149,195,529
	l1i_tlb_refill	7,062,313	3,330,298	8,786,819	3,261,764	4,454,720	3,482,225
CBC 1351MB	L1-dcache-loads	1,407,309,306,177	1,400,027,033,799	1,657,923,261,421	1,655,389,303,882	1,908,470,202,473	1,893,142,065,433
	l1d_cache_refill	184,717,408	203,870,748	208,067,231	251,319,018	260,336,180	269,450,552
	l1d_cache_rd	1,111,510,541,310	1,105,309,635,535	1,316,500,598,610	1,305,963,918,675	1,522,466,958,661	1,506,324,278,000
	l1d_cache_wr	296,582,092,105	295,201,528,790	340,701,228,316	337,784,989,541	385,524,507,545	382,509,095,392
	l1d_cache_wb	56,397,367	69,771,722	68,594,655	87,451,668	73,393,865	73,467,657
	l1d_tlb_refill	79,305,699	140,015,717	81,701,134	151,046,752	88,011,799	151,138,526
	l1d_cache_inval	134,961	963,329	296,344	711,680	266,146	1,160,477
	L1-icache-loads	1,076,040,373,985	1,023,906,553,049	1,267,694,428,954	1,211,358,838,750	1,460,509,579,139	1,384,587,093,125
	l1i_cache_refill	254,823,970	265,420,184	281,077,512	274,144,304	302,300,730	301,006,985
	l1i_tlb_refill	10,370,572	7,030,109	8,624,667	7,506,381	7,023,560	7,942,386

Table D.6: L1 Cache Benchmarks for CBC - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	L1-dcache-loads	80,856,187,461	79,980,518,129	59,088,430,937	57,261,840,381	92,192,085,687	91,063,980,437	66,319,846,180	65,169,964,535	103,669,796,337	102,423,289,744	72,974,967,208	71,600,557,304
	l1d_cache_refill	160,952,001	396,770,928	24,698,710	31,074,803	25,819,819	25,453,429	23,837,338	25,839,876	26,180,053	24,601,691	24,918,042	20,418,342
	l1d_cache_rd	54,894,852,967	54,340,695,761	39,572,313,867	39,867,087,397	62,800,304,464	62,197,179,480	44,454,980,698	43,301,083,220	71,076,224,823	70,510,712,111	49,872,278,228	49,661,555,443
	l1d_cache_wr	25,904,660,299	25,653,721,705	19,222,805,619	18,913,303,105	28,903,550,092	28,561,406,732	21,271,135,500	20,866,946,896	32,153,444,297	31,932,448,869	23,296,332,386	23,017,505,247
	l1d_cache_wb	33,576,787	136,602,384	11,651,287	15,079,213	12,011,175	11,314,470	10,813,718	14,868,068	10,749,550	14,077,110	13,889,732	13,732,953
	l1d_tlb_refill	12,450,769	11,049,640	10,533,866	11,611,188	10,216,264	10,738,401	12,036,831	9,825,274	11,358,009	10,611,669	10,586,749	9,825,845
	l1d_cache_inval	8,675	64,941	5,891	61,959	13,660	68,267	9,104	57,659	46,153	104,005	7,039	57,631
	L1-icache-loads	74,049,222,515	72,547,889,261	39,386,979,932	38,298,035,290	83,615,243,083	83,486,148,068	42,899,349,700	42,086,507,208	94,475,725,113	93,179,077,234	46,499,087,985	45,049,656,237
	l1i_cache_refill	30,142,191	26,551,658	18,916,589	18,012,520	25,447,652	23,216,658	19,476,291	18,541,461	28,628,459	24,991,694	19,970,557	19,703,053
	l1i_tlb_refill	843,853	628,091	515,671	515,808	657,959	608,941	545,241	585,427	757,381	435,867	554,065	374,467
CBC 676MB	L1-dcache-loads	162,044,278,113	161,221,435,014	117,589,426,754	115,871,395,278	183,854,198,403	183,962,129,653	183,962,129,653	129,914,017,984	206,519,918,189	206,741,739,781	146,540,455,243	144,913,577,982
	l1d_cache_refill	61,215,652	85,834,488	47,778,118	53,208,308	54,989,238	56,946,930	56,946,930	51,622,149	70,049,708	59,521,859	45,675,814	54,929,630
	l1d_cache_rd	109,476,765,536	109,167,633,629	79,246,726,657	79,272,166,772	125,898,900,581	125,006,021,145	125,006,021,145	89,048,093,192	142,284,725,137	141,557,226,449	99,494,188,240	98,177,112,754
	l1d_cache_wr	51,825,508,787	51,585,129,644	38,293,154,418	37,892,054,783	57,920,039,605	57,653,925,042	57,653,925,042	42,077,134,482	64,489,041,561	64,211,703,534	46,461,679,360	46,001,360,071
	l1d_cache_wb	26,421,733	40,304,546	23,469,898	32,495,122	22,859,357	28,922,526	28,922,526	35,344,672	28,649,591	29,786,405	21,874,412	29,131,744
	l1d_tlb_refill	25,600,473	26,747,663	22,153,285	24,521,461	22,315,194	23,390,664	23,390,664	22,173,481	23,851,304	23,629,150	21,092,809	22,716,867
	l1d_cache_inval	15,343	139,680	10,491	111,956	18,744	113,011	113,011	118,410	24,220	146,007	15,435	125,860
	L1-icache-loads	148,044,931,741	145,633,047,284	78,475,532,381	76,903,792,875	168,217,359,946	166,781,601,792	166,781,601,792	83,787,647,846	189,456,664,918	187,688,711,160	92,824,418,939	91,187,015,657
	l1i_cache_refill	59,146,105	54,863,559	38,597,703	39,727,168	51,118,173	47,443,737	47,443,737	37,977,465	54,461,998	50,107,931	39,847,290	39,956,425
	l1i_tlb_refill	1,607,361	1,432,010	579,313	1,246,113	1,236,815	788,297	788,297	736,474	933,119	889,006	603,193	1,182,645
CBC 1351MB	L1-dcache-loads	323,723,785,150	322,005,768,755	235,221,689,874	232,847,687,198	367,290,037,911	366,700,553,328	264,079,698,002	261,697,391,546	413,148,698,380	413,178,083,933	292,664,675,936	290,255,016,373
	l1d_cache_refill	110,379,333	112,122,658	132,350,387	107,605,920	138,230,821	119,733,659	98,372,397	97,721,174	121,160,500	119,530,099	89,402,006	1,061,592,309
	l1d_cache_rd	219,359,451,345	219,185,020,582	158,642,651,275	157,864,468,001	251,514,070,029	251,468,292,702	178,959,475,570	178,172,342,792	284,279,267,418	283,566,866,853	199,069,946,707	198,673,283,711
	l1d_cache_wr	103,400,925,819	103,413,465,915	76,721,585,930	76,176,969,636	115,967,587,223	115,434,212,836	84,786,041,804	84,163,016,991	128,923,438,419	128,550,088,012	92,980,611,125	92,331,930,823
	l1d_cache_wb	48,811,499	55,403,720	62,308,924	66,512,084	62,548,113	55,196,084	53,950,306	59,016,579	52,444,431	57,855,066	46,259,140	537,732,524
	l1d_tlb_refill	47,533,095	47,989,544	47,342,858	47,096,228	45,801,689	48,621,850	42,665,785	41,900,461	48,646,968	47,389,371	43,999,520	43,287,441
	l1d_cache_inval	36,751	281,487	23,046	247,774	42,625	290,402	17,649	236,565	61,983	323,377	29,962	253,575
	L1-icache-loads	295,197,504,838	292,538,921,294	157,123,530,241	154,391,025,298	335,731,027,939	333,132,671,825	171,156,422,338	168,310,050,530	376,493,500,325	374,842,855,891	185,539,405,155	182,700,200,590
	l1i_cache_refill	117,847,372	109,551,653	76,826,495	77,515,848	105,354,236	95,210,140	77,861,679	76,490,531	110,636,437	100,571,155	82,489,963	79,224,569
	l1i_tlb_refill	2,474,048	3,053,115	2,236,403	2,356,681	2,809,767	2,207,212	1,169,413	2,377,570	2,821,696	1,654,028	1,259,975	2,384,748

Table D.7: L1 Cache Benchmarks for CBC with Transparent Huge Tables - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	L1-dcache-loads	353,128,896,601	343,421,588,708	415,531,250,608	405,962,198,391	477,602,054,166	465,321,879,638
	l1d_cache_refill	50,359,262	47,171,972	42,845,825	36,850,469	44,395,347	42,305,984
	l1d_cache_rd	278,403,230,476	273,454,671,766	330,076,301,618	320,815,857,641	381,163,603,875	374,259,195,544
	l1d_cache_wr	74,260,829,350	71,232,507,553	85,505,613,656	81,041,661,767	96,509,642,843	92,542,443,729
	l1d_cache_wb	16,580,442	14,266,885	14,577,282	12,214,015	14,244,567	14,687,253
	l1d_tlb_refill	14,164,671	16,820,727	15,078,191	18,776,762	15,773,263	19,819,688
	l1d_cache_inval	44,220	118,061	41,192	174,493	69,354	208,542
	L1-icache-loads	272,204,348,046	260,081,287,168	320,702,524,116	306,914,707,319	367,452,188,910	351,505,821,506
	l1i_cache_refill	43,718,035	31,101,825	41,195,167	35,664,991	44,962,968	39,612,130
	l1i_tlb_refill	1,242,386	930,373	1,191,919	1,066,191	1,819,839	1,035,742
CBC 676MB	L1-dcache-loads	706,092,109,906	689,395,272,892	830,643,420,609	807,850,100,672	957,297,905,264	926,346,075,654
	l1d_cache_refill	87,308,147	77,612,029	108,421,150	89,500,553	121,956,891	126,001,719
	l1d_cache_rd	558,096,683,021	543,753,114,397	660,234,930,171	646,944,794,946	763,628,276,700	744,478,458,096
	l1d_cache_wr	148,871,264,330	142,494,203,971	170,811,290,573	163,608,196,278	193,216,142,650	184,241,011,612
	l1d_cache_wb	29,199,481	27,658,277	34,472,655	30,212,074	32,109,743	32,243,509
	l1d_tlb_refill	26,179,424	34,546,385	29,182,457	38,996,944	32,566,724	41,495,626
	l1d_cache_inval	67,818	303,338	86,334	357,226	133,393	429,290
	L1-icache-loads	546,011,733,180	523,384,421,360	639,615,215,828	612,562,267,250	738,603,157,055	702,157,578,954
	l1i_cache_refill	77,629,295	67,339,887	80,425,144	74,430,074	97,273,874	83,401,328
	l1i_tlb_refill	2,725,296	2,023,355	2,770,792	2,046,956	2,977,109	2,214,998
CBC 1351MB	L1-dcache-loads	1,411,491,893,951	1,375,871,122,343	1,665,181,158,670	1,616,086,953,900	1,909,958,469,811	1,858,308,241,783
	l1d_cache_refill	197,383,997	165,908,305	190,673,336	206,450,913	205,818,891	165,612,377
	l1d_cache_rd	1,113,687,657,223	1,084,733,050,000	1,324,187,618,374	1,281,486,203,001	1,525,040,654,361	1,487,201,771,298
	l1d_cache_wr	296,770,713,270	283,948,921,333	341,823,739,807	324,654,424,332	386,029,557,194	366,730,065,128
	l1d_cache_wb	56,500,597	59,427,891	57,411,177	68,174,474	62,524,971	51,777,744
	l1d_tlb_refill	50,856,706	71,051,123	58,773,421	73,818,308	59,800,406	83,123,795
	l1d_cache_inval	216,766	627,823	243,382	657,080	192,017	872,044
	L1-icache-loads	1,088,485,732,572	1,042,061,964,617	1,280,897,482,826	1,221,364,131,524	1,470,189,300,938	1,403,920,037,244
	l1i_cache_refill	150,643,043	125,888,614	173,773,005	144,895,098	188,557,967	164,273,334
	l1i_tlb_refill	5,822,717	3,923,416	4,316,230	4,261,053	4,499,546	4,535,120

Table D.8: L1 Cache Benchmarks for CBC with Transparent Huge Tables - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	L1-dcache-loads	79,982,328,802	79,512,414,654	58,706,382,481	57,730,835,400	90,916,570,489	90,051,808,817	65,207,224,879	64,655,304,310	102,391,511,471	102,107,012,588	72,359,424,306	71,032,899,578
	l1d_cache_refill	37,557,627	20,243,414	204,422,581	15,546,434	24,060,171	30,165,798	25,409,379	18,955,717	22,313,640	28,751,014	21,020,591	70,190,982
	l1d_cache_rd	54,595,418,775	53,804,797,676	39,424,398,501	38,925,914,736	62,452,725,957	62,282,494,824	44,299,035,058	43,221,669,918	70,616,245,746	70,038,678,236	49,514,943,983	48,440,919,338
	l1d_cache_wr	25,712,260,179	25,072,070,769	18,826,518,911	18,597,689,261	28,827,847,202	28,655,260,146	20,945,853,565	20,695,444,071	31,912,754,295	31,766,666,786	22,988,300,601	22,712,170,890
	l1d_cache_wb	15,901,120	10,078,611	60,594,870	14,920,729	12,122,928	17,250,646	12,181,434	10,774,047	10,008,890	12,690,701	11,764,669	17,048,284
	l1d_tlb_refill	6,005,081	5,406,667	5,164,395	4,446,030	6,143,890	5,595,394	4,540,297	4,648,697	5,878,231	6,190,290	4,959,076	4,657,998
	l1d_cache_inval	8,422	17,677	11,226	13,534	25,403	29,614	4,896	14,901	10,928	32,697	26,475	17,236
	L1-icache-loads	72,485,134,263	72,793,139,824	38,836,236,182	37,296,483,744	82,724,073,870	81,402,655,656	41,753,338,679	41,088,250,496	94,096,812,865	92,639,446,912	45,596,089,180	44,270,425,545
	l1i_cache_refill	14,941,908	9,673,036	8,857,867	7,157,878	11,678,135	10,055,150	8,252,255	7,374,826	11,457,711	10,490,979	8,920,438	7,986,209
	l1i_tlb_refill	471,131	278,511	318,012	352,753	315,916	354,644	296,888	287,697	286,569	362,874	325,030	353,017
CBC 676MB	L1-dcache-loads	160,491,628,370	158,919,485,523	116,111,149,305	115,741,414,680	182,015,797,590	181,421,855,601	129,716,330,673	129,183,375,366	205,089,288,859	203,319,202,912	145,006,286,645	143,449,954,366
	l1d_cache_refill	64,174,282	71,715,801	45,398,420	41,913,020	52,785,826	48,044,090	51,310,916	42,195,186	75,900,841	58,968,514	39,146,777	41,174,862
	l1d_cache_rd	109,153,655,807	107,622,480,306	78,359,036,715	77,646,594,855	125,403,342,275	124,600,910,643	89,193,810,644	88,047,085,099	141,301,448,600	141,481,005,681	99,028,683,720	98,004,037,850
	l1d_cache_wr	51,158,999,334	50,579,054,441	37,834,085,230	37,376,747,027	57,442,844,637	56,941,452,727	42,046,688,133	41,281,692,478	63,948,203,433	63,782,581,143	45,796,121,947	45,121,909,641
	l1d_cache_wb	30,313,404	37,060,106	23,781,096	27,923,198	22,310,295	24,921,855	25,854,153	23,546,639	29,378,342	28,285,143	21,187,230	24,690,668
	l1d_tlb_refill	11,414,425	9,525,043	10,273,300	8,414,093	10,460,473	10,162,278	8,944,985	8,649,024	10,711,088	10,333,187	9,426,162	9,088,824
	l1d_cache_inval	21,589	52,440	10,200	41,553	25,115	55,835	14,678	27,474	19,201	69,482	22,979	33,778
	L1-icache-loads	146,048,320,944	144,019,219,579	76,715,012,113	76,372,714,808	164,746,135,548	164,928,794,925	83,598,153,032	82,832,353,499	186,310,993,983	184,760,339,622	91,892,915,067	90,031,533,650
	l1i_cache_refill	31,992,437	17,308,062	16,462,130	13,891,769	20,957,035	19,207,482	15,686,372	14,349,337	22,690,055	20,897,251	16,057,084	15,474,412
	l1i_tlb_refill	944,308	662,175	440,093	683,257	679,248	709,706	607,824	698,961	572,640	728,710	582,062	534,963
CBC 1351MB	L1-dcache-loads	321,304,776,750	319,442,077,584	235,046,706,440	231,405,546,916	364,961,487,371	362,829,390,290	260,657,290,007	259,865,547,727	409,887,922,234	410,343,144,264	290,227,720,501	288,756,105,735
	l1d_cache_refill	107,690,961	82,606,147	90,811,096	71,993,758	129,114,916	121,525,520	94,627,194	65,072,767	116,577,451	129,366,460	72,525,208	63,019,171
	l1d_cache_rd	218,463,306,134	218,328,703,739	158,720,539,393	155,771,551,823	250,378,543,487	250,140,594,683	177,442,041,251	176,844,784,922	283,013,321,930	282,209,428,903	198,332,840,836	196,746,762,592
	l1d_cache_wr	102,571,635,878	102,209,354,984	76,926,594,340	74,740,137,590	114,858,039,534	114,459,486,642	83,663,419,613	82,696,537,039	127,734,505,403	127,007,711,423	91,900,671,409	90,828,402,332
	l1d_cache_wb	47,745,460	45,648,099	41,573,938	49,243,349	52,830,769	51,174,881	54,600,237	53,627,594	45,469,833	52,405,157	41,675,939	47,015,013
	l1d_tlb_refill	25,906,599	24,318,183	31,344,701	19,366,488	21,832,955	20,431,307	21,233,556	35,172,101	21,394,224	21,293,980	18,984,329	18,735,009
	l1d_cache_inval	76,641	104,076	115,279	51,085	70,761	101,550	18,715	51,246	50,702	130,647	28,386	49,611
	L1-icache-loads	292,294,417,311	290,742,116,146	157,233,263,996	152,273,621,343	330,462,130,444	329,319,597,461	167,738,722,380	165,565,747,575	372,154,728,310	372,752,595,766	183,526,462,726	181,177,692,590
	l1i_cache_refill	71,948,639	54,444,483	66,524,668	30,225,498	43,069,965	37,908,684	30,090,974	28,040,284	45,363,737	39,953,230	31,536,373	29,536,487
	l1i_tlb_refill	1,851,475	1,770,456	2,010,219	1,290,657	1,156,127	1,349,413	846,354	1,023,929	1,441,174	1,435,418	1,216,071	1,473,408

Table D.9: L2 Cache Benchmarks for CBC - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	l2d_cache	203,772,091	283,683,744	263,201,697	252,581,284	249,464,048	291,930,897
	l2d_cache_refill	18,278,947	35,730,792	26,604,084	23,293,726	20,049,029	24,383,669
	l2d_cache_rd	153,987,655	196,237,717	168,328,978	195,597,025	178,733,013	226,479,827
	l2d_cache_wr	73,170,333	64,920,906	69,325,024	84,607,049	72,610,010	85,609,701
	l2d_cache_wb	5,500,701	9,487,198	8,371,095	7,537,080	5,899,094	7,843,331
	l2d_cache_inval	0	0	0	0	0	0
CBC 676MB	l2d_cache	504,325,816	556,978,454	528,897,186	585,090,568	490,169,182	702,717,204
	l2d_cache_refill	53,166,822	51,176,328	58,943,918	55,347,971	42,615,225	75,347,981
	l2d_cache_rd	324,101,881	423,589,888	348,751,267	470,156,464	396,312,347	462,200,297
	l2d_cache_wr	146,276,748	144,485,838	153,085,573	154,497,592	164,493,548	154,517,151
	l2d_cache_wb	11,577,767	16,614,301	11,365,287	17,943,484	12,437,331	17,477,625
	l2d_cache_inval	0	0	0	0	0	0
CBC 1351MB	l2d_cache	909,078,261	1,173,498,913	954,078,080	1,287,614,007	1,110,376,971	1,318,548,168
	l2d_cache_refill	98,917,854	128,293,317	99,020,713	120,025,986	102,803,037	125,496,475
	l2d_cache_rd	639,377,305	817,346,403	683,727,345	926,706,991	735,966,723	937,335,089
	l2d_cache_wr	275,712,265	299,667,971	302,738,189	339,041,100	333,236,243	371,857,734
	l2d_cache_wb	25,702,978	34,188,402	28,036,736	31,165,564	23,513,017	33,782,834
	l2d_cache_inval	0	0	0	0	0	0

Table D.10: L2 Cache Benchmarks for CBC - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	l2d_cache	412,000,073	862,423,046	164,615,795	156,489,760	134,946,984	160,766,034	121,425,339	125,364,349	188,318,591	151,570,688	143,322,880	175,687,297
	l2d_cache_refill	23,816,424	24,437,535	36,076,623	25,087,188	19,127,547	23,566,345	18,796,813	19,793,942	36,182,278	27,019,531	20,538,183	39,880,936
	l2d_cache_rd	213,952,510	454,659,310	81,278,554	93,612,103	122,255,277	111,702,973	87,652,465	111,783,740	102,546,476	108,783,327	93,504,749	87,628,521
	l2d_cache_wr	169,472,910	409,488,981	54,114,475	55,766,148	69,090,854	62,282,388	54,000,605	64,747,275	55,168,829	52,278,090	53,240,316	49,641,210
	l2d_cache_wb	8,815,151	9,421,796	6,500,445	7,153,806	6,072,680	7,320,796	6,133,423	6,443,729	5,926,000	9,559,934	7,240,005	5,943,674
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CBC 676MB	l2d_cache	354,065,786	421,044,500	301,564,015	282,047,158	280,924,695	323,839,168	323,839,168	293,718,163	326,896,421	375,419,873	302,679,755	318,439,025
	l2d_cache_refill	61,688,751	70,780,594	60,678,242	53,497,486	40,086,196	59,236,160	59,236,160	51,850,547	41,282,302	70,040,637	61,590,441	60,774,469
	l2d_cache_rd	202,627,623	219,272,209	157,386,141	192,473,556	187,579,575	219,978,878	219,978,878	210,887,880	194,888,760	206,363,697	158,371,713	207,232,940
	l2d_cache_wr	98,809,710	123,105,904	88,789,738	104,514,503	105,207,453	119,823,608	119,823,608	122,258,467	120,557,328	117,141,515	87,306,628	113,336,515
	l2d_cache_wb	14,010,035	18,498,052	13,737,747	19,987,727	13,758,696	16,394,050	16,394,050	18,472,880	13,582,385	17,097,948	13,496,402	13,188,283
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CBC 1351MB	l2d_cache	612,846,047	659,293,618	529,279,255	610,724,670	649,262,196	590,934,650	569,206,455	599,711,120	633,311,570	647,858,215	558,812,467	2,506,051,149
	l2d_cache_refill	99,660,506	118,647,457	75,455,605	115,552,333	101,632,618	94,700,885	99,586,194	113,811,117	102,266,242	115,338,612	105,659,579	112,177,750
	l2d_cache_rd	415,315,555	440,385,393	378,831,769	382,254,953	385,906,965	423,358,033	368,414,213	407,475,343	422,096,663	411,442,887	356,861,711	1,362,449,228
	l2d_cache_wr	199,721,030	219,823,330	224,691,585	206,366,528	214,346,456	236,972,536	201,300,549	225,149,580	225,455,392	221,101,189	172,124,775	1,159,589,281
	l2d_cache_wb	27,161,102	36,583,461	27,431,481	35,092,493	26,322,305	33,735,921	26,548,496	33,187,613	26,812,862	32,982,138	29,063,509	32,143,400
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Table D.11: L2 Cache Benchmarks for CBC with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	l2d_cache	215,440,676	211,520,206	203,759,764	181,299,560	194,363,751	198,094,982
	l2d_cache_refill	25,517,513	30,188,504	30,790,689	25,647,524	21,804,790	26,195,741
	l2d_cache_rd	133,976,260	156,980,696	113,420,086	186,165,338	124,404,594	148,928,799
	l2d_cache_wr	60,333,042	90,059,798	53,289,688	95,633,331	75,040,350	60,460,759
	l2d_cache_wb	7,483,511	11,132,951	9,639,729	8,785,988	6,933,173	9,516,073
	l2d_cache_inval	0	0	0	0	0	0
CBC 676MB	l2d_cache	386,825,285	367,022,846	450,893,865	405,962,694	510,323,242	471,928,739
	l2d_cache_refill	51,171,710	57,559,786	58,141,343	56,453,040	63,012,359	50,187,598
	l2d_cache_rd	264,328,272	244,836,127	260,837,842	319,018,487	296,671,839	326,747,406
	l2d_cache_wr	132,691,461	115,244,042	164,387,705	158,034,525	173,250,047	181,955,186
	l2d_cache_wb	16,407,993	18,061,327	15,341,779	21,092,082	13,561,189	17,115,694
	l2d_cache_inval	0	0	0	0	0	0
CBC 1351MB	l2d_cache	775,276,840	836,392,669	865,332,244	889,241,634	937,303,541	863,334,345
	l2d_cache_refill	93,374,061	126,440,719	121,154,530	120,913,001	125,548,067	132,415,493
	l2d_cache_rd	557,957,574	521,909,744	517,561,263	587,902,486	544,564,682	557,263,248
	l2d_cache_wr	312,754,353	273,367,374	259,276,627	302,812,142	259,660,501	264,435,413
	l2d_cache_wb	28,297,684	37,096,073	32,995,953	34,808,612	33,300,911	40,432,999
	l2d_cache_inval	0	0	0	0	0	0

Table D.12: L2 Cache Benchmarks for CBC with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	l2d_cache	177,940,570	124,304,770	497,047,003	145,034,692	120,579,723	164,984,674	138,586,779	126,634,124	93,786,297	106,581,147	94,512,270	260,304,157
	l2d_cache_refill	32,147,720	29,809,744	29,141,347	42,876,454	22,218,156	33,333,957	26,928,591	32,338,920	20,453,308	19,878,911	19,756,032	42,472,816
	l2d_cache_rd	86,193,285	77,949,372	282,558,099	59,834,020	66,594,636	69,734,826	62,001,982	52,170,817	60,247,426	79,298,856	56,481,090	106,313,208
	l2d_cache_wr	51,702,807	41,565,185	232,095,366	30,173,907	42,378,935	45,561,482	41,389,961	30,631,148	33,908,501	47,955,803	34,182,013	83,579,303
	l2d_cache_wb	10,812,284	10,304,049	9,912,451	17,431,421	7,637,061	12,528,769	9,196,999	13,422,399	7,358,612	6,836,273	7,416,694	7,884,508
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CBC 676MB	l2d_cache	244,922,680	344,898,556	257,862,036	264,351,134	312,774,041	219,867,854	226,049,243	231,309,085	331,769,593	258,664,017	186,813,326	262,719,930
	l2d_cache_refill	39,928,374	68,103,624	58,895,803	65,329,862	71,937,972	49,066,990	42,270,517	54,012,521	64,844,865	52,186,756	38,653,567	63,787,733
	l2d_cache_rd	167,422,669	161,610,371	155,664,883	141,731,339	130,788,218	178,247,963	118,379,272	131,487,119	152,502,324	146,958,708	168,777,025	163,898,850
	l2d_cache_wr	126,589,341	109,171,912	90,543,291	91,201,517	71,336,187	88,327,759	77,863,474	79,320,508	98,368,776	89,197,442	109,329,906	87,664,464
	l2d_cache_wb	14,696,656	24,206,572	19,700,892	24,470,807	17,210,104	19,375,062	16,552,909	16,906,005	19,728,380	18,786,250	15,015,353	23,832,823
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CBC 1351MB	l2d_cache	559,656,509	509,067,306	498,270,175	512,983,515	634,610,687	574,727,470	450,582,223	438,903,590	583,502,567	592,791,169	440,620,071	421,130,460
	l2d_cache_refill	111,951,069	115,402,835	99,618,696	123,575,341	128,648,502	113,125,027	92,498,190	112,842,418	120,797,759	116,023,971	105,256,352	106,683,749
	l2d_cache_rd	356,592,021	349,099,078	358,819,232	338,252,052	325,899,849	366,796,093	291,923,501	288,288,815	284,873,066	409,417,063	252,513,679	298,991,102
	l2d_cache_wr	204,222,630	207,212,352	211,810,860	190,381,580	198,881,896	231,723,748	186,477,114	177,944,698	196,742,610	246,312,519	171,493,271	200,779,998
	l2d_cache_wb	31,800,780	40,171,975	32,190,201	39,137,245	35,974,573	42,677,664	33,123,554	36,360,712	34,230,257	38,584,571	32,319,854	44,307,018
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Table D.13: Branch Prediction and Speculation Benchmarks for CBC - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	br_pred	65,861,477,673	65,242,570,907	75,482,679,263	74,736,674,659	85,271,018,230	84,232,759,415
	branch-misses	371,269,929	308,569,774	403,478,536	324,794,268	441,837,896	350,326,269
	br_immed_spec	58,811,279,554	58,081,620,690	67,885,487,844	67,161,980,483	77,352,232,818	76,342,244,067
	br_indirect_spec	7,253,061,850	7,164,441,674	7,653,493,918	7,653,946,802	8,144,739,855	8,160,403,347
	br_return_spec	5,682,467,882	5,619,755,717	5,973,509,099	5,979,724,939	6,342,939,583	6,355,996,608
CBC 676MB	br_pred	131,569,364,108	130,931,061,838	151,236,123,988	150,083,695,569	171,000,252,767	169,480,933,253
	branch-misses	733,600,566	626,959,443	785,727,666	666,426,347	970,061,869	712,726,845
	br_immed_spec	117,381,993,979	116,244,743,200	136,017,514,116	134,926,193,211	154,885,648,056	152,514,542,763
	br_indirect_spec	14,389,872,568	14,366,560,276	15,319,730,401	15,324,634,015	16,278,392,202	16,204,917,274
	br_return_spec	11,286,829,476	11,269,211,214	11,966,811,125	11,972,420,769	12,666,158,252	12,628,290,664
CBC 1351MB	br_pred	65,861,477,673	65,242,570,907	75,482,679,263	74,736,674,659	85,271,018,230	84,232,759,415
	branch-misses	371,269,929	308,569,774	403,478,536	324,794,268	441,837,896	350,326,269
	br_immed_spec	58,811,279,554	58,081,620,690	67,885,487,844	67,161,980,483	77,352,232,818	76,342,244,067
	br_indirect_spec	7,253,061,850	7,164,441,674	7,653,493,918	7,653,946,802	8,144,739,855	8,160,403,347
	br_return_spec	5,682,467,882	5,619,755,717	5,973,509,099	5,979,724,939	6,342,939,583	6,355,996,608

Table D.14: Branch Prediction and Speculation Benchmarks for CBC - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	br_pred	38,243,643,579	38,104,773,041	14,915,500,705	14,465,134,258	43,619,472,307	43,307,598,045	15,615,474,019	15,345,913,299	49,088,474,772	49,194,862,434	16,175,998,059	16,154,367,581
	branch-misses	275,058,698	201,447,394	135,625,375	129,128,670	219,789,798	305,952,396	136,876,753	134,620,538	311,996,539	243,348,962	136,188,594	135,581,189
	br_immed_spec	36,445,047,699	36,410,378,500	13,141,319,478	13,109,131,128	41,998,663,339	41,795,242,953	13,789,639,876	13,663,799,083	47,236,894,500	47,094,951,162	14,433,796,493	14,158,037,457
	br_indirect_spec	1,701,977,232	1,718,677,226	1,846,821,521	1,842,834,936	1,748,327,028	1,759,272,021	1,916,005,613	1,906,297,498	1,778,776,494	1,786,280,576	1,985,986,060	1,958,995,072
	br_return_spec	1,089,859,795	1,098,869,043	1,217,181,868	1,218,766,897	1,111,256,398	1,125,099,752	1,281,298,086	1,279,728,231	1,150,080,809	1,151,461,606	1,349,171,228	1,341,671,508
CBC 676MB	br_pred	76,850,503,736	76,724,135,696	29,801,321,067	29,434,922,230	87,086,111,474	87,064,838,143	87,064,838,143	31,146,825,252	97,903,475,280	98,055,731,761	32,131,732,216	32,452,700,825
	branch-misses	548,677,622	407,681,124	272,279,207	265,333,022	605,272,956	447,846,759	447,846,759	270,075,067	664,048,067	496,184,683	268,366,640	271,874,038
	br_immed_spec	73,034,999,777	72,668,396,892	26,418,686,928	26,042,053,211	83,782,680,021	83,736,358,443	83,736,358,443	27,175,553,544	94,614,001,772	94,151,731,173	28,432,195,770	28,272,960,664
	br_indirect_spec	3,416,966,337	3,476,139,916	3,718,955,622	3,666,854,239	3,464,622,403	3,470,705,102	3,470,705,102	3,775,403,498	3,548,552,774	3,532,217,677	3,924,662,518	3,901,348,130
	br_return_spec	2,163,470,390	2,229,191,756	2,464,413,999	2,423,406,656	2,216,126,565	2,223,999,753	2,223,999,753	2,535,268,813	2,287,418,671	2,280,785,757	2,674,210,526	2,664,399,084
CBC 1351MB	br_pred	153,577,490,076	153,338,698,763	59,396,556,371	59,612,013,113	174,275,967,402	173,976,772,018	62,396,672,334	62,209,781,042	195,913,480,569	195,907,133,730	64,610,514,903	64,461,368,488
	branch-misses	1,024,675,878	857,560,077	535,097,089	539,066,922	1,134,747,667	876,361,995	542,214,438	540,507,128	1,060,051,129	939,017,438	542,644,398	536,320,057
	br_immed_spec	146,447,525,552	145,939,008,656	52,640,294,136	52,124,369,223	167,609,622,377	167,593,170,932	54,928,788,872	54,373,983,299	189,153,277,448	188,776,997,175	56,807,792,155	56,768,426,453
	br_indirect_spec	6,879,060,788	6,906,909,138	7,423,869,242	7,339,417,267	6,934,846,848	7,001,870,794	7,642,753,486	7,573,055,770	7,056,820,405	7,068,895,285	7,858,581,469	7,823,922,906
	br_return_spec	4,364,986,172	4,414,837,999	4,904,550,026	4,861,866,111	4,420,881,479	4,492,017,580	5,128,335,398	5,092,949,703	4,543,331,936	4,576,761,613	5,361,110,801	5,341,890,452

Table D.15: Branch Prediction and Speculation Benchmarks for CBC with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	br_pred	65,986,757,257	65,654,349,037	75,658,147,948	75,610,227,900	85,212,984,820	84,850,618,319
	branch-misses	401,756,664	352,494,418	475,302,789	394,348,282	453,691,407	432,612,672
	br_immed_spec	58,876,796,738	58,373,198,821	68,194,097,408	67,794,759,600	77,219,229,909	76,391,822,494
	br_indirect_spec	7,183,537,906	7,225,990,734	7,638,641,553	7,676,641,803	8,060,870,580	8,149,286,006
	br_return_spec	5,634,439,709	5,612,263,060	5,961,989,551	5,963,122,154	6,277,877,942	6,261,119,473
CBC 676MB	br_pred	131,711,861,128	131,239,577,256	151,124,779,890	150,429,956,612	171,034,348,603	169,596,281,551
	branch-misses	841,122,280	727,124,178	870,224,012	785,813,076	1,027,008,384	855,275,234
	br_immed_spec	117,698,261,917	117,238,853,207	136,256,085,375	135,933,368,083	154,871,956,979	154,395,042,865
	br_indirect_spec	14,316,561,464	14,383,199,554	15,269,148,112	15,384,820,855	16,169,140,563	16,487,237,321
	br_return_spec	11,215,758,627	11,231,467,141	11,923,520,147	11,954,736,173	12,589,119,853	12,669,632,632
CBC 1351MB	br_pred	263,103,656,051	263,152,670,831	302,711,838,713	301,958,005,770	341,291,090,758	341,322,644,834
	branch-misses	1,536,323,270	1,438,749,590	1,707,001,520	1,604,215,361	2,030,449,892	1,826,413,134
	br_immed_spec	234,573,654,568	233,987,153,393	272,297,426,482	271,435,176,422	309,094,674,101	307,975,449,235
	br_indirect_spec	28,532,392,942	28,841,827,067	30,417,603,277	30,967,382,080	32,190,351,277	32,890,390,619
	br_return_spec	22,366,125,641	22,440,782,709	23,754,116,349	23,891,721,194	25,055,604,752	25,258,338,092

Table D.16: Branch Prediction and Speculation Benchmarks for CBC with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CBC 115MB	br_pred	37,673,689,994	37,391,459,858	14,838,050,619	14,396,679,671	43,306,843,594	42,906,283,495	15,213,796,056	14,897,427,245	48,475,689,751	48,098,961,748	15,663,475,979	15,487,885,735
	branch-misses	201,728,321	258,549,253	134,646,519	132,952,681	238,390,622	213,184,331	133,350,304	131,786,938	329,884,631	231,369,378	133,909,749	131,019,659
	br_immed_spec	36,213,398,728	35,958,312,854	12,584,144,476	12,350,752,077	41,593,256,251	40,982,601,162	13,329,191,128	13,118,565,897	46,985,278,848	46,892,537,719	13,992,967,269	13,763,855,433
	br_indirect_spec	1,652,733,585	1,600,135,315	1,737,757,053	1,710,248,138	1,668,132,442	1,652,484,582	1,835,742,680	1,801,779,159	1,714,201,786	1,710,458,004	1,913,515,147	1,886,736,321
	br_return_spec	1,042,862,735	1,006,697,975	1,143,058,189	1,122,136,022	1,060,820,102	1,057,397,265	1,231,109,855	1,205,170,636	1,099,593,708	1,104,497,795	1,302,165,005	1,288,605,897
CBC 676MB	br_pred	75,984,852,945	75,611,936,592	29,225,128,788	28,911,666,799	86,447,565,190	85,976,212,793	30,182,547,753	29,910,578,545	97,477,332,894	96,775,227,564	31,412,103,404	31,779,472,186
	branch-misses	412,803,363	405,472,593	267,274,061	263,265,384	437,192,611	444,552,231	264,803,402	263,009,987	504,754,073	466,795,219	264,245,924	266,770,741
	br_immed_spec	72,936,982,290	72,075,189,301	25,513,037,563	24,988,981,021	83,101,579,397	83,129,041,024	26,902,503,367	27,039,958,246	93,642,087,051	93,664,789,191	27,983,523,862	27,371,808,156
	br_indirect_spec	3,363,246,064	3,261,513,422	3,545,046,317	3,460,607,296	3,349,079,602	3,372,261,615	3,707,705,361	3,724,285,098	3,373,940,529	3,433,960,531	3,824,685,716	3,725,557,410
	br_return_spec	2,118,565,135	2,055,598,911	2,334,611,539	2,278,988,645	2,121,385,134	2,153,856,538	2,483,522,305	2,491,380,212	2,160,914,222	2,209,046,463	2,599,487,066	2,534,040,422
CBC 1351MB	br_pred	151,925,266,856	150,963,594,039	59,274,664,589	58,072,953,702	173,044,909,837	172,377,136,789	60,571,785,899	60,296,572,760	194,288,460,516	194,326,978,657	63,235,854,540	62,792,363,378
	branch-misses	858,498,646	822,177,573	539,296,348	529,624,034	901,026,528	871,849,036	530,669,424	531,560,747	944,909,344	931,762,269	528,936,891	528,865,964
	br_immed_spec	145,571,661,097	145,398,724,107	52,469,935,137	50,771,589,086	166,071,721,608	166,330,825,851	53,666,089,806	53,801,379,162	187,747,582,771	187,415,766,295	55,729,557,686	56,037,406,302
	br_indirect_spec	6,721,426,172	6,751,093,492	7,377,331,468	7,089,532,461	6,645,564,231	6,755,509,269	7,397,925,283	7,401,612,097	6,802,220,688	6,814,835,848	7,610,092,427	7,632,763,488
	br_return_spec	4,242,586,628	4,269,042,452	4,885,114,139	4,666,589,845	4,210,111,138	4,303,412,948	4,949,761,510	4,942,743,843	4,356,345,838	4,379,564,574	5,168,852,760	5,171,848,307

Appendix E

Tables - CFB

Table E.1: General Benchmarks for CFB - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	context-switches	21,881	17,877	25,607	20,809	29,055	23,914
	page-faults	121,178	121,193	121,179	121,193	121,179	121,193
	cycles	457,780,712,889	463,936,668,270	531,745,425,046	532,664,883,426	611,080,633,622	621,218,641,882
	instructions	834,001,492,479	849,723,064,955	976,943,364,231	975,087,938,709	1,129,198,939,770	1,130,898,778,890
	time elapsed (s)	256.259337038	185.130138381	297.672539101	213.904373635	342.056962439	246.669707870
	cache-misses	46,697,693	75,017,220	59,571,871	53,773,568	59,600,572	59,707,345
	mem_access	357,991,648,716	356,977,401,090	414,532,219,879	411,646,595,185	477,586,246,268	467,766,765,788
	mem_access_rd	283,226,049,296	278,220,544,756	329,270,132,058	322,103,807,201	381,178,019,993	377,027,629,562
	mem_access_wr	74,560,933,268	72,842,849,246	85,205,618,396	83,486,637,396	96,383,616,985	95,260,025,147
CFB 676 MB	context-switches	43,675	34,570	50,913	42,204	58,841	46,240
	page-faults	242,234	242,249	242,234	242,250	242,233	242,251
	cycles	912,439,421,259	920,601,063,030	1,063,524,504,777	1,073,189,486,776	1,222,366,315,451	1,247,451,976,885
	instructions	1,665,773,497,130	1,688,388,983,808	1,954,421,022,205	1,948,363,765,618	2,257,527,449,780	2,271,346,428,359
	time elapsed (s)	510.813887170	370.539225830	595.384778984	427.053468478	684.299101101	481.187987545
	cache-misses	93,033,844	137,246,961	126,647,334	95,776,716	111,656,285	110,048,127
	mem_access	714,188,252,060	713,710,632,648	829,472,074,845	815,595,878,214	954,847,030,795	952,404,506,107
	mem_access_rd	565,562,548,072	555,083,295,675	659,177,395,186	656,483,514,118	762,348,119,936	754,697,917,028
	mem_access_wr	148,552,847,897	145,350,957,848	170,622,815,257	169,545,089,880	192,881,184,083	191,285,454,553
CFB 1135 MB	context-switches	94,734	68,881	101,319	81,018	121,551	92,292
	page-faults	491,986	484,408	484,342	484,756	490,775	484,453
	cycles	1,828,257,872,242	1,815,243,813,389	2,121,716,020,106	2,152,736,465,930	2,442,741,810,052	2,468,513,026,879
	instructions	3,337,656,281,003	3,334,560,119,778	3,912,378,145,070	3,917,889,834,522	4,518,822,254,841	4,511,469,573,881
	time elapsed (s)	1025.129579868	718.091889415	1187.694609088	840.032197423	1367.646723223	962.502770714
	cache-misses	208,346,480	275,773,231	201,905,852	235,412,476	225,816,039	249,667,798
	mem_access	1,429,920,395,300	1,409,172,664,654	1,658,340,160,764	1,643,982,359,553	1,911,438,826,362	1,890,753,886,453
	mem_access_rd	1,132,148,935,964	1,120,180,541,600	1,317,458,138,507	1,307,853,536,761	1,525,237,461,082	1,511,325,870,663
	mem_access_wr	298,297,682,731	292,880,974,867	340,899,451,469	338,341,343,176	386,628,415,239	381,981,511,351

Table E.2: General Benchmarks for CFB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	context-switches	7,649	6,280	3,327	2,825	6,363	4,774	3,736	3,103	7,184	5,305	3,910	3,287
	page-faults	121,179	121,194	121,178	121,194	121,180	121,194	121,178	121,194	121,178	121,194	121,178	121,194
	cycles	117,475,484,449	115,263,714,787	70,347,548,585	70,108,789,013	127,155,729,076	125,398,723,939	73,988,390,997	73,676,210,658	142,885,185,937	141,797,181,652	81,263,252,606	81,243,153,827
	instructions	224,708,669,421	222,432,680,227	138,531,299,086	134,153,278,659	249,969,079,927	247,914,838,310	135,421,279,305	131,735,706,003	281,338,358,232	281,275,133,737	148,912,758,700	146,156,370,878
	time elapsed (s)	66.015936855	47.442285849	39.421945083	31.169860365	71.223016204	52.260361463	41.473384377	33.566019650	80.050792139	57.967762267	45.533133009	36.055751984
	cache-misses	33,652,185	41,037,455	31,676,358	58,133,350	29,767,128	29,592,243	24,153,386	22,760,545	26,620,516	30,661,390	24,670,374	26,004,981
	mem_access	78,571,508,885	77,790,503,807	68,212,691,985	68,248,343,798	91,814,770,439	91,395,691,994	65,770,593,812	65,527,645,868	103,632,378,428	102,540,589,952	73,145,526,514	72,480,436,382
	mem_access_rd	52,827,658,915	52,797,122,098	42,079,514,074	42,085,024,781	62,679,394,259	62,055,569,699	44,699,839,240	43,511,093,817	70,755,517,484	70,695,746,379	49,819,775,893	49,144,075,549
	mem_access_wr	25,901,196,964	25,863,769,668	25,231,221,178	25,102,652,087	28,840,145,743	28,565,922,739	21,209,398,425	20,844,396,428	32,002,911,866	31,950,798,127	23,123,538,697	22,793,295,134
CFB 676 MB	context-switches	18,842	12,455	6,984	6,083	12,714	14,051	7,095	5,956	13,766	10,595	7,901	6,432
	page-faults	242,235	242,253	242,234	242,257	242,236	242,262	242,233	242,248	242,235	242,251	242,233	242,460
	cycles	234,322,163,558	233,033,009,256	139,320,099,145	140,536,985,608	253,053,513,761	252,296,861,013	149,143,568,524	147,930,523,787	285,079,163,326	283,278,858,644	162,518,386,248	161,413,308,870
	instructions	448,352,389,356	448,298,552,876	277,190,164,802	271,191,658,268	499,637,612,014	496,851,585,308	270,061,698,649	266,595,235,929	563,160,466,658	561,311,444,962	297,469,345,022	290,625,166,715
	time elapsed (s)	131.793892624	95.026160855	78.057085572	62.349343145	141.726663731	105.301819740	83.541362746	66.373757099	159.648644369	115.867433586	91.056964071	71.848633670
	cache-misses	74,272,370	86,214,168	63,074,620	96,026,216	61,319,345	77,434,863	55,513,550	56,615,726	67,912,393	57,963,446	54,120,479	79,388,042
	mem_access	157,369,568,431	156,759,174,296	135,436,168,014	136,555,560,785	183,995,855,729	184,066,452,324	131,908,502,627	130,188,776,278	206,610,634,743	206,072,045,249	146,246,900,371	144,825,511,513
	mem_access_rd	105,752,643,102	105,715,853,769	85,204,355,992	84,823,091,029	125,518,880,637	124,887,834,891	89,460,254,818	89,006,792,786	141,750,062,622	140,938,844,920	99,665,578,458	97,847,516,950
	mem_access_wr	51,744,043,580	51,795,128,859	50,890,811,096	50,618,182,110	57,924,605,045	57,554,100,408	42,339,955,813	42,054,997,434	64,231,131,439	63,823,447,029	46,434,683,522	45,916,651,538
CFB 1135 MB	context-switches	30,657	25,010	15,526	19,265	30,471	20,112	14,355	12,756	27,323	21,318	15,653	13,037
	page-faults	484,343	484,442	493,605	497,606	484,343	485,657	485,741	485,715	488,384	485,709	484,341	484,358
	cycles	465,232,012,349	465,269,485,571	279,966,540,818	286,472,192,377	505,381,364,683	505,199,593,932	298,261,470,068	297,673,625,855	571,366,528,095	569,290,265,535	325,262,525,441	325,480,221,185
	instructions	897,112,479,211	897,151,540,664	555,447,347,749	549,939,308,496	997,501,512,454	995,523,209,762	539,765,863,847	530,117,513,623	1,125,819,140,875	1,125,710,674,376	593,751,437,111	582,326,779,056
	time elapsed (s)	261.099449153	190.419035609	157.315037742	128.800342201	284.325143194	208.617189718	167.099851594	134.486336305	319.947526992	232.059428883	182.183381353	144.190278442
	cache-misses	115,841,268	202,466,208	104,258,376	164,767,220	122,989,190	116,606,654	108,512,656	125,368,227	128,218,569	130,498,544	109,995,322	112,506,990
	mem_access	315,277,004,764	313,980,974,765	272,631,385,820	274,562,952,075	367,710,043,063	366,906,433,006	264,171,903,846	262,013,305,815	413,180,587,187	412,142,766,232	292,909,615,268	290,221,917,901
	mem_access_rd	211,388,486,185	211,364,613,803	170,543,088,476	171,835,971,116	251,500,303,345	250,959,135,401	178,738,160,775	177,511,021,067	283,813,506,929	283,729,251,844	199,189,072,504	198,170,833,179
	mem_access_wr	103,707,824,010	103,529,724,631	101,829,299,018	102,769,373,983	115,726,416,690	115,571,965,763	84,855,908,609	83,734,608,081	128,717,001,299	128,576,491,126	92,904,196,035	92,171,486,137

Table E.3: General Benchmarks for CFB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	context-switches	23,945	14,767	25,439	18,909	29,150	21,508
	page-faults	5,588	5,621	4,692	4,755	5,475	5,604
	cycles	459,377,633,947	439,201,938,268	531,383,428,079	530,287,008,339	611,852,992,744	607,340,249,591
	instructions	833,013,467,399	804,736,236,662	975,238,642,988	944,664,281,286	1,127,623,823,487	1,090,251,726,821
	time elapsed (s)	257.307095935	167.881162161	297.475026396	200.673207300	342.496024575	229.355343567
	cache-misses	42,589,150	69,148,400	39,814,244	44,330,730	40,678,663	39,920,335
	mem_access	359,440,474,432	348,200,706,104	415,506,711,224	403,997,352,869	478,605,066,844	464,707,209,046
	mem_access_rd	285,016,660,119	277,269,132,400	330,292,856,242	322,384,922,355	381,794,440,112	371,772,152,153
	mem_access_wr	74,935,882,423	71,072,282,646	85,395,299,225	81,565,039,372	96,608,047,502	91,827,772,717
CFB 676 MB	context-switches	48,063	31,013	50,777	38,538	58,627	45,628
	page-faults	43,766	5,722	20,331	5,978	13,067	6,155
	cycles	918,130,030,925	880,978,232,737	1,062,774,880,511	1,066,467,102,012	1,224,946,253,681	1,215,017,803,735
	instructions	1,663,493,807,487	1,610,110,478,013	1,953,667,242,150	1,900,606,642,480	2,255,092,774,088	2,182,115,320,291
	time elapsed (s)	514.255189546	345.526468867	594.903842094	409.480793211	685.658124013	474.682745015
	cache-misses	84,860,398	116,341,098	94,822,630	84,318,893	102,403,449	94,670,574
	mem_access	717,578,282,751	697,630,120,994	830,134,212,567	807,638,159,655	956,992,410,585	932,953,750,135
	mem_access_rd	568,125,546,846	554,411,184,947	659,806,289,403	646,338,891,569	763,184,160,154	740,271,716,960
	mem_access_wr	149,375,030,660	142,237,843,364	170,556,242,406	163,565,001,117	193,019,023,918	183,092,500,518
CFB 1135 MB	context-switches	96,388	71,673	104,834	83,844	114,465	95,865
	page-faults	6,532	11,201	12,455	6,869	8,423	7,141
	cycles	1,837,469,987,792	1,770,078,862,620	2,147,004,052,970	2,123,622,039,120	2,465,007,323,443	2,432,076,574,196
	instructions	3,331,052,763,659	3,230,868,471,488	3,909,306,573,202	3,790,831,997,026	4,510,556,458,506	4,372,692,773,021
	time elapsed (s)	1030.341909162	725.536361433	1202.151778022	858.709237507	1380.654801512	981.515399906
	cache-misses	161,732,647	316,186,333	2,182,490,284	175,863,714	178,536,361	175,688,425
	mem_access	1,439,126,660,979	1,390,152,773,998	1,668,274,923,983	1,611,672,865,065	1,913,018,402,409	1,852,053,739,985
	mem_access_rd	1,139,817,310,240	1,111,479,717,300	1,324,472,728,773	1,298,376,867,330	1,526,535,891,136	1,485,333,971,854
	mem_access_wr	300,101,645,062	285,454,554,936	344,614,153,629	328,258,076,194	386,569,046,918	367,004,947,546

Table E.4: General Benchmarks for CFB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	context-switches	5,756	4,366	3,255	2,787	6,103	4,849	3,515	2,941	6,693	5,348	3,988	3,606
	page-faults	5,539	5,650	5,572	4,707	4,789	4,755	5,555	5,683	4,772	4,930	5,602	5,586
	cycles	114,597,822,155	113,323,708,791	68,468,085,862	68,221,574,006	126,445,487,804	123,576,563,572	72,688,788,993	72,371,806,267	141,028,942,737	141,418,424,929	80,061,952,461	81,277,569,552
	instructions	222,221,340,841	219,805,860,894	135,771,521,890	132,253,212,375	246,431,081,364	245,527,684,876	133,017,080,301	130,558,410,713	280,669,727,685	280,703,668,997	147,444,031,483	145,070,262,019
	time elapsed (s)	64.182417239	46.460328300	38.393643677	30.493919368	70.815748920	51.552220335	40.727156819	32.655626293	79.000973090	57.583440612	44.859594571	35.977268967
	cache-misses	29,041,627	35,153,243	40,769,923	42,376,462	279,350,821	31,175,741	17,565,757	16,253,515	24,919,177	34,989,813	26,312,100	35,669,352
	mem_access	77,691,249,028	77,229,254,467	67,589,407,801	67,037,835,058	91,168,272,794	90,404,793,741	65,432,991,267	63,987,725,988	102,502,554,702	101,909,622,826	72,834,453,952	70,705,366,306
	mem_access_rd	52,528,233,579	52,086,198,439	42,290,438,883	42,331,590,794	62,527,921,577	62,147,418,913	44,608,279,573	44,039,745,473	70,888,340,239	70,758,284,336	49,736,083,845	49,386,734,843
	mem_access_wr	25,553,605,260	25,343,267,536	25,178,602,629	24,915,334,630	28,618,483,214	28,530,274,106	21,002,127,658	20,608,586,376	31,927,559,054	31,825,459,051	23,027,802,480	22,701,053,011
CFB 676 MB	context-switches	14,684	9,391	6,494	5,496	11,951	10,398	7,534	6,696	13,592	10,748	7,104	6,550
	page-faults	6,107	6,701	5,722	6,156	6,218	6,305	6,030	6,057	6,233	6,243	6,228	6,148
	cycles	229,605,726,229	228,640,959,560	136,224,439,208	138,849,579,433	250,183,116,972	249,447,895,775	146,735,874,359	146,366,095,439	282,588,407,779	281,358,936,232	160,560,401,887	160,886,213,834
	instructions	445,115,852,269	443,193,445,751	274,845,501,557	268,792,130,666	495,410,775,075	492,990,478,223	265,230,497,110	261,824,035,847	561,388,030,942	558,698,032,105	292,988,294,910	288,337,076,372
	time elapsed (s)	128.855743220	92.868034020	76.294314949	60.881311823	140.109083911	102.970353208	82.250713641	66.038231444	158.250002448	115.066315375	89.917538643	71.742655074
	cache-misses	46,186,688	64,308,186	48,335,488	84,897,705	56,020,050	46,742,876	39,552,130	46,772,031	60,029,904	54,210,033	46,784,426	38,503,460
	mem_access	156,274,006,067	154,450,110,508	132,679,859,564	136,263,148,635	182,546,901,297	181,881,863,345	131,113,818,499	130,514,176,718	204,530,407,339	204,351,555,115	145,510,573,106	143,505,034,312
	mem_access_rd	105,129,316,368	104,641,865,285	83,761,101,628	84,886,281,493	125,446,160,983	124,672,212,583	88,609,004,522	88,425,600,313	141,963,747,945	141,148,446,019	99,139,271,227	98,117,078,425
	mem_access_wr	51,248,996,758	50,883,339,134	50,099,267,963	50,335,751,243	57,403,794,426	57,147,074,933	41,781,078,995	41,322,756,562	64,008,514,219	63,558,918,541	45,953,336,409	45,463,990,428
CFB 1135 MB	context-switches	30,025	16,233	15,716	10,981	24,130	19,017	13,156	11,824	27,408	21,483	15,206	12,849
	page-faults	100,347	6,757	31,442	6,792	16,409	6,903	7,367	6,711	6,727	7,046	7,017	6,791
	cycles	460,169,963,421	455,422,057,525	274,221,717,852	278,681,017,845	500,199,686,605	498,542,884,684	293,213,704,470	290,517,048,591	565,350,465,171	564,899,221,734	320,991,506,540	319,576,590,379
	instructions	891,173,683,021	883,289,354,250	547,916,471,184	538,008,936,006	991,675,190,379	986,369,706,628	532,755,532,484	524,224,055,918	1,121,194,613,524	1,120,420,446,255	585,918,232,161	577,095,538,080
	time elapsed (s)	259.681166212	184.918991103	154.222899297	122.290872537	280.096042371	205.995128092	164.225509658	130.313302787	316.582940543	229.969170403	179.778067158	141.345806414
	cache-misses	93,311,810	123,834,983	122,784,171	135,512,983	99,781,988	97,995,454	81,295,252	72,195,354	113,242,498	132,016,687	75,610,761	87,167,673
	mem_access	313,247,471,576	311,071,939,495	270,146,550,722	271,492,205,568	364,624,253,849	364,248,550,869	261,755,170,919	260,049,260,243	411,150,311,083	409,916,507,174	290,683,105,375	287,951,571,951
	mem_access_rd	210,539,571,418	209,052,723,169	167,781,706,615	169,817,483,542	250,443,596,788	250,393,349,278	177,743,315,876	176,743,071,070	283,097,398,525	282,829,257,713	198,018,242,710	197,835,214,555
	mem_access_wr	102,738,573,619	101,582,826,639	99,745,880,374	100,631,165,859	114,739,076,472	114,545,409,165	83,596,463,422	82,627,956,853	127,597,030,390	127,462,398,368	91,758,794,965	91,599,363,680

Table E.5: L1 Cache Benchmarks for CFB - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	L1-dcache-loads	358,238,155,684	362,784,733,072	413,788,438,318	410,057,834,266	477,938,440,684	475,368,442,375
	l1d_cache_refill	43,382,345	76,595,336	65,012,169	57,339,097	59,265,384	62,965,494
	l1d_cache_rd	282,964,224,724	269,692,478,000	329,353,857,596	326,436,924,607	381,353,681,165	383,221,097,454
	l1d_cache_wr	74,525,960,436	71,431,377,135	85,295,086,892	83,566,413,361	96,464,568,087	96,281,995,706
	l1d_cache_wb	12,744,196	30,102,597	21,141,167	19,752,531	16,677,026	18,181,240
	l1d_tlb_refill	17,711,412	32,997,031	23,926,939	34,106,407	20,981,147	37,207,438
	l1d_cache_inval	41,081	16,263,005	43,238	259,422	55,549	280,606
	L1-icache-loads	276,535,801,296	260,843,945,909	317,065,724,803	300,804,044,822	366,269,618,990	345,113,394,301
	l1i_cache_refill	65,298,880	62,773,025	70,240,207	68,815,922	74,812,215	72,337,921
	l1i_tlb_refill	1,791,461	1,763,820	2,380,847	1,776,639	2,085,928	1,922,320
CFB 676MB	L1-dcache-loads	713,674,840,061	723,226,716,046	829,230,452,427	814,768,059,990	954,067,239,215	953,684,504,010
	l1d_cache_refill	98,232,566	133,701,984	124,377,211	98,278,545	112,684,056	113,569,442
	l1d_cache_rd	565,512,836,628	549,842,146,070	659,229,139,072	659,553,213,753	761,992,636,568	748,009,064,117
	l1d_cache_wr	148,492,901,382	144,033,202,047	170,740,549,325	170,299,323,869	192,976,592,855	189,874,161,982
	l1d_cache_wb	31,414,035	60,505,837	39,292,355	31,206,532	31,893,258	36,458,717
	l1d_tlb_refill	41,896,531	66,643,386	43,166,478	70,286,290	45,379,155	84,423,402
	l1d_cache_inval	90,156	30,487,410	84,034	516,906	100,559	570,524
	L1-icache-loads	549,369,679,627	522,241,060,327	635,889,421,033	593,667,828,655	731,720,981,323	695,967,165,021
	l1i_cache_refill	132,645,638	127,652,444	141,201,439	133,768,363	151,200,982	145,160,116
	l1i_tlb_refill	3,472,979	3,646,907	4,076,537	3,339,471	4,077,066	3,889,465
CFB 1351MB	L1-dcache-loads	1,429,518,269,018	1,409,302,806,371	1,658,149,729,547	1,646,910,132,641	1,911,226,842,137	1,887,480,829,634
	l1d_cache_refill	211,069,264	276,942,181	203,231,978	238,609,778	226,706,554	248,435,379
	l1d_cache_rd	1,132,229,654,699	1,119,542,655,891	1,317,510,157,717	1,305,734,828,262	1,524,970,278,069	1,511,854,470,158
	l1d_cache_wr	298,134,381,982	292,871,762,085	340,865,995,429	337,696,707,093	386,567,894,249	382,120,404,584
	l1d_cache_wb	69,552,058	122,661,974	60,695,194	75,423,571	67,099,917	76,813,059
	l1d_tlb_refill	85,839,493	125,928,915	85,821,890	133,463,815	89,214,653	158,622,579
	l1d_cache_inval	139,413	68,561,164	163,327	1,009,143	251,939	1,202,386
	L1-icache-loads	1,101,645,147,410	1,020,535,369,515	1,268,392,787,476	1,204,745,217,643	1,467,015,009,479	1,376,686,965,311
	l1i_cache_refill	286,560,802	247,235,790	281,990,103	266,739,783	315,082,422	291,291,582
	l1i_tlb_refill	8,495,530	7,084,715	15,126,766	7,374,517	10,019,381	7,883,210

Table E.6: L1 Cache Benchmarks for CFB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	L1-dcache-loads	79,041,592,356	77,435,271,331	68,320,307,306	68,146,536,146	91,852,018,284	91,082,717,386	66,133,433,712	65,116,572,015	103,452,615,761	102,428,042,071	72,962,971,820	73,216,419,768
	l1d_cache_refill	30,310,888	41,950,729	30,615,034	57,829,931	28,125,201	27,999,416	22,223,199	23,551,755	27,461,678	29,776,788	22,788,827	24,775,105
	l1d_cache_rd	52,924,483,394	52,826,415,178	42,228,119,119	41,849,865,259	62,876,437,607	62,233,583,174	44,712,063,895	43,266,467,143	71,044,317,387	70,634,862,083	49,851,416,169	48,820,799,874
	l1d_cache_wr	25,824,634,618	25,914,796,309	25,170,893,467	24,984,525,239	28,985,175,187	28,602,514,497	21,319,616,942	20,839,347,423	32,146,580,563	32,071,987,379	23,363,550,729	22,633,968,346
	l1d_cache_wb	13,468,764	21,845,569	13,481,974	33,608,885	10,985,194	12,686,730	11,144,248	12,120,225	12,157,310	14,342,071	11,348,051	13,181,161
	l1d_tlb_refill	14,035,410	15,266,233	10,119,441	11,385,304	10,274,748	10,420,034	9,807,066	9,631,814	10,735,239	10,658,862	10,573,104	10,058,769
	l1d_cache_inval	10,788	8,398,783	4,478	16,816,191	12,071	70,890	7,503	54,979	21,578	81,126	8,505	61,495
	L1-icache-loads	78,634,382,371	76,934,981,434	44,389,276,912	43,788,981,162	83,983,766,556	82,908,161,827	42,687,193,973	42,084,854,611	94,558,614,086	93,166,102,456	46,341,750,303	46,136,608,597
	l1i_cache_refill	31,681,811	28,996,119	19,980,657	19,335,455	26,428,788	23,652,241	20,079,685	19,025,367	28,040,287	25,038,337	20,533,647	20,040,421
	l1i_tlb_refill	688,530	688,856	575,380	572,599	710,661	614,477	601,135	405,394	742,592	634,381	335,443	443,452
CFB 676MB	L1-dcache-loads	157,708,381,906	157,189,142,389	136,318,860,165	135,737,389,751	184,124,043,818	183,876,829,889	131,846,894,931	130,362,012,878	206,932,986,752	205,952,026,317	146,518,803,004	145,133,227,677
	l1d_cache_refill	73,082,229	87,601,987	61,664,371	95,197,889	59,779,469	76,907,779	54,255,730	54,256,703	68,801,237	58,789,142	52,105,917	78,440,708
	l1d_cache_rd	105,846,445,521	105,295,201,063	84,855,060,539	85,744,165,083	125,687,064,241	124,667,367,419	89,700,564,009	88,967,158,999	141,936,669,021	142,107,499,860	99,727,050,687	99,505,157,182
	l1d_cache_wr	51,837,767,233	51,669,870,747	50,797,391,342	50,898,239,861	57,974,642,658	57,681,975,962	42,430,899,140	42,001,572,812	64,297,477,481	64,120,988,456	46,499,086,173	46,190,924,116
	l1d_cache_wb	32,761,572	47,117,555	30,933,172	63,665,689	24,760,168	34,341,121	27,413,251	35,568,305	25,735,837	29,899,114	25,755,081	39,507,858
	l1d_tlb_refill	29,041,851	28,892,312	22,652,716	25,433,703	26,947,949	22,117,383	21,475,625	21,370,573	20,983,849	23,915,802	21,532,019	21,772,707
	l1d_cache_inval	12,887	17,053,912	9,152	35,262,119	34,919	131,960	10,954	121,944	29,320	155,662	14,556	124,265
	L1-icache-loads	157,380,377,303	155,638,957,062	88,307,131,937	87,003,084,186	167,452,441,048	167,088,802,045	85,512,255,420	83,696,495,333	188,169,390,989	187,122,071,522	92,828,060,206	91,575,708,118
	l1i_cache_refill	67,193,470	57,105,266	40,847,906	41,869,151	53,351,971	51,348,930	39,960,795	38,352,142	54,478,867	50,105,817	41,476,367	39,670,985
	l1i_tlb_refill	1,457,432	1,554,707	658,563	1,173,015	1,451,296	1,334,778	601,829	791,653	1,400,787	874,819	693,622	1,334,748
CFB 1351MB	L1-dcache-loads	315,055,180,782	314,316,904,083	272,246,013,758	274,459,503,006	367,872,531,208	367,162,792,143	263,776,399,841	261,593,844,993	413,232,941,496	412,616,811,067	292,450,819,503	290,568,240,480
	l1d_cache_refill	115,825,308	203,573,313	108,872,737	162,486,908	120,841,217	122,171,409	111,294,123	126,955,506	129,114,873	130,498,188	109,330,405	112,008,426
	l1d_cache_rd	211,375,030,125	211,126,901,254	170,496,772,506	172,572,408,507	251,459,312,094	250,870,617,287	179,083,719,496	178,463,327,795	284,231,199,125	283,518,371,987	198,988,631,330	198,736,149,214
	l1d_cache_wr	103,548,157,113	103,548,880,756	101,744,256,416	103,024,277,360	115,974,677,573	115,678,790,341	85,034,499,272	84,643,406,282	128,894,083,216	128,436,197,909	93,027,194,390	92,549,740,608
	l1d_cache_wb	49,755,277	126,192,423	51,442,072	95,485,798	56,187,445	64,566,635	50,012,463	58,758,714	49,270,473	66,499,334	56,071,317	64,080,225
	l1d_tlb_refill	49,952,197	54,702,874	49,016,900	56,338,181	49,095,441	47,941,001	42,334,683	46,323,260	46,691,594	50,651,771	42,949,256	48,206,149
	l1d_cache_inval	42,941	70,149,267	29,892	37,561,935	67,301	331,948	18,948	250,078	36,753	301,809	22,493	258,439
	L1-icache-loads	311,202,068,011	310,306,798,777	177,403,224,202	176,550,504,840	334,044,654,056	333,093,077,610	171,231,377,640	168,408,481,573	376,751,860,739	374,737,108,355	185,908,289,450	182,615,874,033
	l1i_cache_refill	119,893,769	114,090,378	84,655,230	98,604,350	111,075,141	93,939,444	77,771,913	77,872,264	106,113,121	100,068,013	80,719,189	79,214,616
	l1i_tlb_refill	3,233,548	2,496,152	1,444,259	3,007,587	1,996,462	1,790,721	2,295,634	2,451,480	2,726,033	2,418,610	1,275,886	2,278,613

Table E.7: L1 Cache Benchmarks for CFB with Transparent Huge Tables - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	L1-dcache-loads	359,749,839,894	347,893,006,348	415,230,845,110	403,823,492,324	478,770,497,657	464,100,803,281
	l1d_cache_refill	42,128,714	68,204,405	42,721,460	43,571,110	40,418,939	39,288,067
	l1d_cache_rd	285,105,172,706	276,642,060,569	329,769,866,473	322,470,938,037	381,688,676,872	371,902,664,815
	l1d_cache_wr	74,836,040,263	70,824,642,915	85,341,409,037	81,380,810,320	96,726,603,611	92,099,475,330
	l1d_cache_wb	14,098,892	31,704,409	13,774,860	13,583,064	11,186,060	12,369,924
	l1d_tlb_refill	13,413,370	13,768,958	14,574,517	18,922,540	15,318,879	18,896,228
	l1d_cache_inval	48,530	23,089,618	50,367	152,519	51,317	152,376
	L1-icache-loads	281,084,153,949	267,231,852,223	319,980,059,852	305,973,133,971	369,429,117,251	350,854,804,077
	l1i_cache_refill	42,745,761	29,305,994	40,744,029	34,897,515	45,229,936	40,190,771
	l1i_tlb_refill	1,247,022	970,844	1,205,521	990,182	1,353,063	1,137,166
CFB 676MB	L1-dcache-loads	717,423,499,244	696,859,910,828	829,712,916,250	810,658,983,085	957,119,009,199	932,171,379,942
	l1d_cache_refill	85,813,191	116,056,920	95,701,672	83,873,845	104,218,124	94,304,039
	l1d_cache_rd	567,604,446,418	555,315,769,732	659,739,455,257	640,729,746,042	763,583,710,462	743,495,956,041
	l1d_cache_wr	149,303,680,939	142,479,079,155	170,700,763,817	162,884,039,354	193,225,374,293	183,728,711,915
	l1d_cache_wb	29,031,506	52,732,546	36,463,013	28,929,534	28,344,108	33,388,735
	l1d_tlb_refill	27,187,646	30,986,684	28,748,933	42,204,264	31,808,120	39,617,959
	l1d_cache_inval	91,973	25,605,294	90,529	327,664	120,012	434,960
	L1-icache-loads	557,255,040,495	537,052,086,595	638,764,667,047	612,302,943,987	738,625,055,618	705,039,885,706
	l1i_cache_refill	97,108,703	60,532,097	85,277,332	70,727,361	92,311,749	81,926,859
	l1i_tlb_refill	2,763,750	1,917,591	2,604,504	1,914,622	2,623,969	2,305,549
CFB 1351MB	L1-dcache-loads	1,439,432,113,098	1,393,853,634,421	1,668,120,825,675	1,606,999,444,026	1,913,232,708,522	1,858,753,717,894
	l1d_cache_refill	162,751,545	318,127,055	2,185,370,203	173,574,722	173,840,534	175,505,350
	l1d_cache_rd	1,139,682,669,544	1,119,173,412,221	1,323,754,307,217	1,293,228,333,097	1,527,159,902,948	1,489,567,675,848
	l1d_cache_wr	300,190,335,523	286,169,451,765	344,284,653,628	326,959,137,918	386,598,375,958	367,801,722,049
	l1d_cache_wb	46,775,917	136,497,129	77,490,772	58,785,094	47,011,763	50,793,165
	l1d_tlb_refill	49,094,289	61,384,991	62,323,622	76,966,336	69,931,646	84,201,974
	l1d_cache_inval	180,494	94,114,379	140,298	804,550	425,070	885,292
	L1-icache-loads	1,126,068,242,519	1,074,756,038,892	1,298,113,721,525	1,214,725,047,180	1,479,602,570,936	1,399,618,097,978
	l1i_cache_refill	161,065,619	139,564,422	167,313,666	153,418,347	181,370,676	173,827,757
	l1i_tlb_refill	4,582,345	4,127,095	4,844,443	4,493,870	5,396,035	4,457,145

Table E.8: L1 Cache Benchmarks for CFB with Transparent Huge Tables - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	L1-dcache-loads	77,818,695,780	77,027,309,012	66,555,125,906	66,290,587,368	91,721,072,305	89,939,229,037	65,174,169,365	64,773,946,875	102,353,896,804	102,178,319,863	72,366,683,677	72,480,348,955
	l1d_cache_refill	27,969,505	34,774,690	43,896,264	41,015,916	278,503,005	30,568,987	17,718,965	16,947,750	24,754,788	34,054,231	27,313,284	35,272,024
	l1d_cache_rd	52,240,495,122	51,827,262,875	41,802,953,093	42,237,557,073	62,625,330,620	62,390,348,218	44,542,712,604	43,533,109,174	70,797,435,509	70,261,319,976	49,420,912,013	48,997,691,745
	l1d_cache_wr	25,476,258,083	25,236,199,959	25,313,087,079	25,260,866,777	28,693,860,853	28,642,866,433	21,005,506,810	20,698,052,374	32,001,486,284	31,681,548,527	23,028,894,003	22,471,658,655
	l1d_cache_wb	12,552,769	19,550,266	19,810,937	29,057,497	15,222,432	15,752,479	10,972,612	9,982,651	11,497,312	13,602,657	13,579,232	15,188,488
	l1d_tlb_refill	5,151,699	6,145,243	4,695,529	4,129,849	5,362,282	4,976,233	5,136,066	4,584,429	5,484,571	5,599,273	5,167,964	4,737,822
	l1d_cache_inval	14,210	8,279,075	5,575	19,492,354	7,732	27,467	4,467	14,075	26,021	29,211	6,416	15,150
	L1-icache-loads	76,801,819,302	76,275,171,476	43,498,813,271	42,039,545,633	83,022,177,501	81,676,917,870	42,183,662,190	41,403,644,955	92,472,252,966	92,458,059,355	45,521,704,685	44,635,080,492
	l1i_cache_refill	10,529,886	9,981,217	8,033,347	7,083,456	10,659,779	9,519,063	8,066,553	7,725,215	11,389,493	10,330,381	8,914,373	8,755,036
	l1i_tlb_refill	347,584	323,802	305,027	364,009	287,190	385,478	303,526	288,610	376,915	293,494	318,591	378,866
CFB 676MB	L1-dcache-loads	156,466,995,646	155,960,835,075	134,697,125,129	134,809,103,277	182,089,074,485	182,879,823,330	130,281,491,557	129,476,424,788	205,431,067,768	204,948,730,809	144,566,206,630	143,883,845,842
	l1d_cache_refill	46,608,950	61,993,057	47,450,753	84,490,619	56,441,566	44,673,774	42,826,412	47,724,400	55,517,696	51,306,996	49,586,338	31,912,256
	l1d_cache_rd	104,703,668,038	104,123,972,221	84,860,778,686	84,034,561,593	125,277,526,103	124,738,558,021	89,339,008,124	88,138,335,262	141,396,884,335	140,772,026,498	98,899,170,411	98,715,577,378
	l1d_cache_wr	51,066,016,882	50,628,040,243	50,469,421,684	50,194,394,561	57,474,211,538	57,090,891,061	42,021,481,259	41,341,383,159	63,872,273,143	63,522,212,936	46,016,963,687	45,518,515,117
	l1d_cache_wb	22,556,137	39,405,319	24,498,515	57,563,018	23,122,946	24,308,277	25,229,287	28,128,860	21,130,669	23,297,372	23,895,703	24,327,918
	l1d_tlb_refill	11,465,369	10,544,197	10,718,030	12,602,287	10,543,758	10,473,732	9,209,246	9,102,247	11,051,209	11,203,239	9,815,947	11,062,884
	l1d_cache_inval	14,871	17,473,839	6,124	35,411,173	20,183	64,769	25,274	38,831	20,990	51,838	23,052	34,138
	L1-icache-loads	154,487,212,589	154,137,587,601	86,480,295,617	85,619,145,085	165,294,723,183	165,479,392,456	84,446,383,528	82,580,992,359	186,175,918,571	186,076,233,140	91,591,911,547	89,807,245,280
	l1i_cache_refill	29,629,246	19,915,183	16,177,181	13,478,438	20,802,547	19,793,896	15,833,813	15,315,398	23,306,909	20,793,037	15,236,123	14,980,165
	l1i_tlb_refill	864,056	728,984	570,548	527,523	539,179	717,464	592,766	687,504	747,234	735,208	550,963	703,499
CFB 1351MB	L1-dcache-loads	312,787,368,548	309,971,474,553	270,235,241,466	269,898,032,451	364,692,585,690	362,893,909,914	260,680,103,111	260,005,290,746	411,142,336,223	410,515,458,677	288,999,767,770	287,129,527,177
	l1d_cache_refill	95,947,272	123,382,599	122,274,512	135,736,450	98,546,419	96,659,780	83,288,682	75,424,779	113,937,708	129,821,029	76,945,562	80,839,444
	l1d_cache_rd	210,189,631,222	208,768,193,341	169,693,829,489	169,109,376,597	250,329,599,038	249,843,118,311	177,920,817,098	176,395,287,628	282,643,830,236	281,925,313,196	198,120,109,272	196,723,468,036
	l1d_cache_wr	102,551,950,586	101,661,989,948	100,558,169,598	100,545,064,603	114,741,445,511	114,592,988,702	83,634,847,679	82,731,801,221	127,372,666,322	127,076,523,104	91,798,182,445	91,323,523,994
	l1d_cache_wb	45,081,836	74,116,005	55,797,420	79,372,706	45,453,187	55,595,806	45,869,909	53,713,729	43,445,681	60,569,229	43,104,518	54,518,939
	l1d_tlb_refill	34,367,510	19,656,785	19,303,863	18,311,824	21,145,687	20,191,723	17,260,904	18,439,685	21,389,936	22,478,033	18,617,999	17,030,974
	l1d_cache_inval	48,953	30,449,621	45,745	38,588,568	37,816	97,404	42,127	50,907	41,448	140,637	20,874	57,079
	L1-icache-loads	309,524,545,772	306,192,587,779	174,571,985,658	171,909,687,166	330,137,831,730	329,164,738,977	167,740,888,861	166,841,210,138	373,460,122,626	372,585,678,054	182,366,001,986	179,996,935,210
	l1i_cache_refill	68,101,760	33,524,370	33,162,155	26,481,408	42,357,105	36,998,376	29,013,766	27,563,167	45,451,741	40,316,097	31,482,801	29,480,617
	l1i_tlb_refill	1,776,423	1,001,592	1,233,960	1,365,681	1,345,115	1,054,529	1,097,439	1,297,345	1,468,276	1,135,047	872,095	1,354,497

Table E.9: L2 Cache Benchmarks for CFB - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	l2d_cache	220,173,098	318,297,471	244,758,474	268,267,886	306,670,527	289,212,793
	l2d_cache_refill	23,485,282	18,937,989	20,739,490	26,208,006	39,207,263	23,343,834
	l2d_cache_rd	158,740,816	245,188,832	160,017,688	180,713,140	179,946,617	213,126,650
	l2d_cache_wr	66,644,134	93,162,355	69,185,518	65,518,317	84,701,604	83,234,001
	l2d_cache_wb	7,162,186	5,820,770	5,816,555	8,195,700	5,346,902	7,164,808
	l2d_cache_inval	0	0	0	0	0	0
CFB 676MB	l2d_cache	435,751,029	641,370,560	522,125,814	587,812,381	501,973,610	601,912,766
	l2d_cache_refill	42,952,009	46,792,946	47,217,683	66,014,453	45,533,092	53,258,163
	l2d_cache_rd	337,356,788	455,549,045	334,456,868	428,377,674	333,762,338	472,105,677
	l2d_cache_wr	142,031,373	162,403,471	155,623,898	145,748,127	138,380,442	161,029,370
	l2d_cache_wb	13,521,015	14,542,767	14,176,531	13,916,074	13,419,217	16,928,617
	l2d_cache_inval	0	0	0	0	0	0
CFB 1351MB	l2d_cache	1,023,023,369	1,260,103,835	941,774,567	1,167,942,675	1,043,308,782	1,338,889,874
	l2d_cache_refill	112,895,915	90,849,689	81,238,579	103,498,855	98,839,112	130,473,326
	l2d_cache_rd	678,463,495	886,648,895	687,325,487	888,556,302	706,540,322	967,012,664
	l2d_cache_wr	290,343,655	332,177,205	282,545,798	344,598,617	297,192,832	338,232,147
	l2d_cache_wb	27,910,421	28,301,794	24,427,118	33,840,919	23,199,162	34,919,799
	l2d_cache_inval	0	0	0	0	0	0

Table E.10: L2 Cache Benchmarks for CFB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	l2d_cache	166,750,472	167,197,233	138,940,331	182,648,322	200,011,537	158,925,952	172,844,885	118,431,275	130,205,985	161,415,669	179,657,231	161,240,163
	l2d_cache_refill	26,069,426	23,067,288	19,824,136	19,716,958	40,640,921	23,035,162	37,452,805	20,108,171	18,868,690	28,254,875	34,426,418	30,895,553
	l2d_cache_rd	129,976,154	101,330,038	124,494,044	119,803,727	96,616,810	110,620,344	86,749,520	104,128,960	99,126,377	95,575,823	89,122,213	103,316,527
	l2d_cache_wr	63,627,997	54,734,310	80,829,595	76,359,387	57,514,819	66,192,795	52,811,955	62,448,351	53,424,795	58,532,842	53,878,171	60,434,793
	l2d_cache_wb	9,073,309	7,677,363	6,517,095	6,627,722	6,075,373	7,511,371	6,132,258	6,559,393	5,847,587	10,092,921	6,596,373	11,898,258
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CFB 676MB	l2d_cache	349,880,527	340,160,173	340,962,149	324,568,898	278,914,182	326,715,341	310,403,036	303,899,573	303,950,125	293,979,845	326,642,934	340,742,506
	l2d_cache_refill	47,888,083	43,730,104	61,511,891	40,035,341	39,430,960	45,190,961	58,685,880	54,995,872	39,245,006	49,901,462	60,722,881	52,585,079
	l2d_cache_rd	233,422,515	213,734,947	185,946,254	189,680,127	196,377,577	235,905,873	165,067,795	215,249,429	240,662,362	244,660,049	176,859,989	215,911,318
	l2d_cache_wr	117,690,847	110,060,758	106,356,886	120,849,186	110,622,996	131,662,766	101,542,909	123,332,258	132,361,548	130,161,174	105,986,409	136,886,869
	l2d_cache_wb	16,900,623	14,304,544	14,057,412	13,000,796	12,978,296	15,938,831	13,093,543	19,743,369	13,077,224	17,342,898	13,187,989	18,988,687
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CFB 1351MB	l2d_cache	623,537,178	754,113,476	554,358,540	646,420,887	604,434,375	631,816,361	556,547,525	653,487,888	618,649,153	705,091,051	526,534,991	629,479,118
	l2d_cache_refill	100,117,684	92,384,606	99,571,818	86,101,478	84,684,557	117,326,954	97,992,437	119,080,802	96,416,517	125,345,606	79,801,229	120,350,386
	l2d_cache_rd	439,640,429	462,167,764	393,873,668	418,216,561	407,875,274	393,476,879	334,862,692	400,502,411	404,246,368	424,220,831	373,366,567	375,215,181
	l2d_cache_wr	214,289,354	265,599,633	204,926,896	232,362,892	219,407,355	211,131,514	196,410,653	229,711,947	226,152,539	222,235,458	213,817,347	211,518,296
	l2d_cache_wb	26,022,705	29,229,758	27,298,499	29,020,256	28,877,003	33,085,536	26,230,878	34,457,455	24,908,340	36,762,390	27,251,073	34,120,199
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Table E.11: L2 Cache Benchmarks for CFB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	l2d_cache	238,934,459	228,809,158	175,962,697	203,316,363	176,695,390	195,970,837
	l2d_cache_refill	44,096,852	21,714,507	20,633,052	30,984,870	19,912,985	25,607,568
	l2d_cache_rd	132,553,370	147,644,544	116,543,851	128,621,643	141,203,321	137,964,928
	l2d_cache_wr	70,026,105	85,805,587	57,368,697	61,247,288	74,004,861	66,200,899
	l2d_cache_wb	8,020,516	7,655,437	7,352,891	10,815,002	5,962,693	8,678,411
	l2d_cache_inval	0	0	0	0	0	0
CFB 676MB	l2d_cache	395,832,562	396,033,785	439,414,932	435,455,235	401,255,692	492,970,963
	l2d_cache_refill	47,091,914	43,221,085	58,953,976	71,474,151	40,328,117	75,209,520
	l2d_cache_rd	280,795,161	266,839,130	248,520,669	275,888,773	299,111,028	289,781,830
	l2d_cache_wr	129,446,690	162,309,140	135,272,259	134,847,502	169,786,313	134,054,105
	l2d_cache_wb	14,948,812	14,527,592	18,697,493	16,818,280	13,758,089	18,603,098
	l2d_cache_inval	0	0	0	0	0	0
CFB 1351MB	l2d_cache	798,897,269	1,003,399,304	4,816,281,566	856,981,040	833,668,068	852,830,845
	l2d_cache_refill	124,322,936	96,026,685	116,324,418	131,532,471	120,877,604	121,174,740
	l2d_cache_rd	467,739,192	635,219,283	2,509,296,838	599,127,682	512,466,011	580,054,784
	l2d_cache_wr	225,047,774	396,510,971	2,244,339,300	304,799,207	253,078,317	247,465,456
	l2d_cache_wb	35,233,307	32,157,002	31,397,387	41,563,175	32,048,583	32,371,156
	l2d_cache_inval	0	0	0	0	0	0

Table E.12: L2 Cache Benchmarks for CFB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	l2d_cache	156,328,187	165,057,339	145,916,584	151,386,624	676,677,256	177,565,546	94,769,020	108,907,719	168,093,680	155,725,582	107,212,803	173,488,535
	l2d_cache_refill	30,894,495	30,716,407	20,844,461	25,025,709	40,658,400	40,621,103	24,660,142	30,429,521	41,816,750	31,098,663	19,116,736	37,902,935
	l2d_cache_rd	68,674,436	73,671,430	76,320,163	70,771,468	311,959,832	74,245,629	53,949,981	53,493,565	63,057,540	86,552,019	61,500,235	84,378,210
	l2d_cache_wr	42,653,673	47,856,141	66,758,454	50,157,183	285,899,044	50,763,435	35,678,922	30,776,733	35,880,369	52,705,516	41,334,648	48,810,684
	l2d_cache_wb	9,563,433	9,758,188	7,232,001	8,431,366	6,659,481	8,103,763	7,145,835	12,524,288	7,280,079	12,059,560	7,316,881	15,911,743
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CFB 676MB	l2d_cache	250,335,818	299,007,766	305,490,924	272,181,795	258,427,042	270,848,462	187,378,552	239,054,449	353,170,059	282,025,401	200,958,424	274,451,431
	l2d_cache_refill	52,498,583	57,744,037	69,604,425	42,420,062	54,727,227	66,433,967	39,125,071	51,519,752	79,505,222	61,301,412	39,082,103	70,537,739
	l2d_cache_rd	185,653,431	145,756,575	180,027,157	138,143,307	131,301,473	150,232,199	156,119,783	143,331,837	133,850,758	193,770,705	116,558,139	128,458,379
	l2d_cache_wr	85,706,389	95,391,707	128,865,264	99,382,325	80,363,066	96,880,151	100,400,752	90,870,906	83,103,586	106,556,138	86,150,431	70,756,621
	l2d_cache_wb	18,503,758	20,793,496	27,707,754	14,105,481	15,794,697	17,700,296	14,247,856	19,924,486	19,142,622	22,673,968	15,001,261	25,577,504
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0
CFB 1351MB	l2d_cache	474,067,841	505,047,889	555,941,536	492,833,462	553,002,732	550,147,504	460,460,774	423,864,048	475,135,062	619,609,824	407,494,178	581,000,160
	l2d_cache_refill	90,047,563	95,024,349	108,372,291	89,311,848	123,477,416	120,530,462	104,801,830	103,479,751	91,846,924	129,510,816	88,751,435	143,978,108
	l2d_cache_rd	355,677,991	282,661,549	341,915,708	270,333,950	303,413,838	336,756,196	282,009,539	301,345,660	345,109,110	353,217,350	278,924,371	266,565,869
	l2d_cache_wr	202,425,694	188,211,898	248,862,613	192,777,882	183,316,985	205,045,794	180,272,310	196,497,696	207,242,389	211,385,516	170,549,470	192,645,478
	l2d_cache_wb	31,562,075	32,862,963	31,614,649	32,581,075	35,105,424	45,919,844	31,745,697	41,608,737	30,660,039	40,499,238	33,441,577	43,247,834
	l2d_cache_inval	0	0	0	0	0	0	0	0	0	0	0	0

Table E.13: Branch Prediction and Speculation Benchmarks for CFB - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	br_pred	72,746,497,910	72,874,261,816	75,380,906,483	74,848,684,713	85,427,186,663	84,558,254,912
	branch-misses	362,908,724	245,773,333	388,095,363	327,716,769	467,727,062	363,413,856
	br_immed_spec	65,574,184,774	64,461,196,858	67,962,533,024	67,612,729,996	77,489,342,133	75,369,404,874
	br_indirect_spec	7,183,276,478	7,199,984,671	7,673,406,819	7,682,842,894	8,186,454,533	8,081,790,357
	br_return_spec	5,627,688,958	5,644,043,746	5,995,620,618	6,003,472,903	6,372,511,808	6,297,383,531
CFB 676MB	br_pred	144,931,450,874	145,401,733,407	151,116,507,829	149,306,789,008	170,463,881,896	170,017,405,631
	branch-misses	700,248,003	482,479,967	854,542,484	664,298,662	935,056,419	728,143,654
	br_immed_spec	130,788,280,658	128,634,231,843	136,154,759,585	134,558,140,161	154,587,321,390	153,306,250,407
	br_indirect_spec	14,438,362,149	14,340,370,443	15,319,511,033	15,318,315,157	16,267,992,569	16,288,683,891
	br_return_spec	11,307,803,648	11,234,631,194	11,970,470,522	11,965,255,318	12,654,371,336	12,685,766,475
CFB 1351MB	br_pred	290,971,765,910	286,470,467,859	302,269,896,954	299,947,902,682	342,044,640,642	337,699,751,747
	branch-misses	1,413,499,061	976,469,036	1,663,966,041	1,332,516,187	1,943,400,491	1,388,366,669
	br_immed_spec	262,260,712,754	257,749,668,818	271,994,018,564	269,526,220,435	309,624,575,935	305,878,234,930
	br_indirect_spec	28,889,856,315	28,660,235,483	30,691,884,739	30,704,809,209	32,512,997,849	32,543,001,349
	br_return_spec	22,630,613,456	22,464,149,987	23,971,947,589	23,999,513,096	25,312,106,713	25,337,360,632

Table E.14: Branch Prediction and Speculation Benchmarks for CFB - Compiler Optimizations combined with Software Optimizations

	Benchmarks	**128 Bits Key**				**192 Bits Key**				**256 Bits Key**			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	br_pred	42,728,276,012	42,179,457,043	16,445,864,728	16,093,408,179	43,665,558,033	43,280,346,372	15,573,245,896	15,534,181,689	48,939,038,953	48,805,196,784	16,208,837,306	16,085,811,991
	branch-misses	275,259,993	162,365,935	135,559,351	129,976,212	304,486,600	235,989,154	136,779,804	135,388,687	293,522,155	233,648,541	136,019,585	135,560,169
	br_immed_spec	40,801,202,911	40,562,541,947	14,722,379,692	14,599,965,305	41,983,004,319	41,833,727,183	13,853,929,471	13,651,162,922	47,390,874,635	47,106,709,179	14,461,570,029	14,335,555,370
	br_indirect_spec	1,733,878,185	1,741,160,375	1,858,254,733	1,836,823,802	1,767,274,570	1,760,761,521	1,930,860,358	1,906,206,126	1,779,516,437	1,777,357,397	1,993,181,227	1,961,655,219
	br_return_spec	1,106,394,565	1,111,709,548	1,226,167,985	1,214,117,510	1,130,000,764	1,123,913,290	1,302,094,657	1,279,760,516	1,143,150,578	1,144,591,000	1,357,869,554	1,328,049,436
CFB 676MB	br_pred	85,183,817,709	85,127,180,969	32,822,522,954	32,570,864,869	87,291,117,955	86,825,138,525	31,156,588,743	31,369,570,391	97,784,597,122	97,711,855,172	32,493,029,259	32,396,798,704
	branch-misses	501,550,998	370,920,773	271,434,254	266,132,100	490,164,825	439,525,514	271,599,952	271,659,166	464,242,461	472,551,491	271,461,419	268,534,936
	br_immed_spec	81,786,505,536	81,282,375,593	29,280,907,851	29,091,308,449	83,814,251,286	83,741,803,329	27,468,210,380	27,167,371,701	94,529,643,241	94,527,400,276	28,495,915,148	28,386,572,324
	br_indirect_spec	3,454,099,148	3,492,091,849	3,699,836,659	3,672,818,760	3,448,101,794	3,490,837,344	3,820,047,081	3,774,140,562	3,519,535,817	3,540,383,436	3,916,828,376	3,917,371,892
	br_return_spec	2,196,152,404	2,225,090,766	2,443,266,210	2,434,111,039	2,197,997,103	2,238,848,703	2,573,041,886	2,535,664,487	2,266,420,308	2,289,933,279	2,670,408,746	2,669,174,099
CFB 1351MB	br_pred	170,322,283,906	170,647,666,269	65,845,678,878	66,405,344,531	174,381,064,571	174,085,184,404	62,254,987,243	61,985,276,462	195,448,358,498	195,845,806,187	64,782,561,544	64,408,418,192
	branch-misses	550,896,032	581,663,965	547,539,157	549,984,164	907,654,910	902,647,982	541,985,599	538,333,865	1,051,018,580	959,060,450	541,309,037	537,435,882
	br_immed_spec	163,537,258,435	162,993,411,964	58,434,360,926	58,726,328,573	167,477,960,715	167,339,745,647	54,900,582,547	54,707,919,253	188,918,581,324	188,694,748,421	57,295,369,187	56,944,934,288
	br_indirect_spec	6,906,112,804	6,970,677,367	7,419,643,328	7,512,037,765	6,912,144,508	6,986,788,466	7,644,131,750	7,614,349,759	7,034,463,307	7,070,068,008	7,901,471,303	7,841,808,548
	br_return_spec	4,398,548,462	4,446,377,581	4,929,403,280	5,011,708,412	4,426,728,411	4,467,074,116	5,129,525,755	5,112,831,558	4,529,681,531	4,582,466,219	5,388,760,737	5,347,146,996

Table E.15: Branch Prediction and Speculation Benchmarks for CFB with Transparent Huge Pages - Software Optimizations

	Benchmarks	**128 Bits Key**		**192 Bits Key**		**256 Bits Key**	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	br_pred	73,012,864,737	72,029,449,731	75,449,327,050	75,371,001,290	85,500,575,528	84,925,996,938
	branch-misses	401,742,525	330,823,077	439,151,998	398,688,134	534,837,879	438,281,942
	br_immed_spec	66,079,936,197	64,887,575,039	68,037,291,384	67,771,404,250	77,468,164,257	76,802,989,695
	br_indirect_spec	7,183,091,287	7,209,724,980	7,606,500,214	7,716,941,461	8,080,188,206	8,190,942,883
	br_return_spec	5,635,048,986	5,603,127,488	5,933,197,039	5,969,372,511	6,286,038,946	6,284,468,576
CFB 676MB	br_pred	145,634,256,351	144,642,537,030	151,314,374,739	151,208,132,372	170,842,351,871	169,930,149,569
	branch-misses	806,962,087	654,920,749	886,870,721	817,797,590	1,023,114,344	893,461,585
	br_immed_spec	131,373,339,756	130,529,977,881	136,017,785,117	135,576,171,295	154,981,701,034	154,349,693,742
	br_indirect_spec	14,378,239,263	14,586,910,134	15,175,866,568	15,450,019,341	16,154,546,041	16,370,812,313
	br_return_spec	11,263,674,214	11,273,662,715	11,843,255,102	11,916,747,076	12,574,908,528	12,670,322,718
CFB 1351MB	br_pred	292,100,908,409	290,377,169,026	305,469,710,352	300,951,321,608	341,409,665,816	339,765,770,270
	branch-misses	1,611,597,396	1,329,583,368	1,920,598,617	1,623,216,137	1,886,668,580	1,712,148,275
	br_immed_spec	263,743,564,156	259,945,996,004	275,018,481,411	269,809,425,730	309,418,903,968	307,039,872,581
	br_indirect_spec	28,516,790,497	29,058,047,627	30,390,891,374	30,691,415,707	32,229,502,097	32,844,681,440
	br_return_spec	22,349,614,125	22,464,176,311	23,740,801,645	23,870,933,653	25,088,029,928	25,234,464,443

Table E.16: Branch Prediction and Speculation Benchmarks for CFB with Transparent Huge Pages - Compiler Optimizations combined with Software Optimizations

	Benchmarks	128 Bits Key				192 Bits Key				256 Bits Key			
		O2		O3		O2		O3		O2		O3	
		Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP	Normal	OpenMP
CFB 115MB	br_pred	42,172,123,895	41,716,429,142	16,022,683,100	15,927,531,666	42,838,880,703	42,923,185,602	14,989,065,598	14,813,615,743	48,660,251,404	48,711,550,458	15,761,057,090	15,854,234,862
	branch-misses	157,396,541	158,963,574	133,790,000	133,133,604	255,721,798	213,386,784	132,389,883	130,805,811	234,545,706	235,100,809	134,301,710	132,794,129
	br_immed_spec	40,290,950,599	40,287,944,260	14,478,782,746	14,047,622,917	41,476,012,531	41,022,788,531	13,348,518,356	13,205,589,782	46,925,875,920	46,507,922,908	13,905,986,404	13,712,182,668
	br_indirect_spec	1,627,218,918	1,638,445,917	1,813,880,389	1,747,852,245	1,668,184,689	1,654,397,666	1,841,482,618	1,815,402,310	1,705,140,903	1,682,467,634	1,898,561,496	1,855,676,820
	br_return_spec	1,017,623,083	1,032,841,861	1,190,452,182	1,152,059,952	1,057,846,355	1,057,804,613	1,232,691,525	1,213,681,302	1,095,640,782	1,077,325,519	1,292,697,606	1,251,846,159
CFB 676MB	br_pred	84,483,932,441	84,180,844,751	32,066,707,088	31,775,977,120	86,367,828,451	85,971,187,865	30,068,719,287	30,030,848,185	97,279,430,270	96,963,786,575	31,425,296,920	31,749,877,323
	branch-misses	276,309,529	289,713,654	266,060,614	264,903,537	451,136,721	434,718,992	264,032,111	264,539,711	474,478,411	473,304,690	265,356,000	266,088,090
	br_immed_spec	81,419,148,275	80,118,859,486	28,456,736,346	28,557,160,945	83,115,199,592	82,769,568,508	27,000,017,553	26,976,373,840	93,354,714,368	93,366,311,774	28,075,529,126	27,527,557,988
	br_indirect_spec	3,373,877,336	3,212,680,984	3,561,710,220	3,562,887,870	3,352,392,215	3,368,053,095	3,722,552,301	3,690,103,883	3,327,220,530	3,345,022,747	3,835,227,255	3,753,460,797
	br_return_spec	2,122,669,205	2,029,047,701	2,355,456,699	2,341,340,783	2,124,511,481	2,137,556,902	2,490,100,322	2,463,386,092	2,139,587,004	2,159,204,409	2,602,479,459	2,555,265,356
CFB 1351MB	br_pred	169,165,954,287	167,754,560,685	64,247,258,944	63,795,139,281	172,903,943,655	172,359,973,378	60,538,400,297	60,218,074,358	194,549,461,220	194,539,259,089	63,010,244,670	63,409,931,183
	branch-misses	539,901,579	547,612,678	533,133,124	531,578,325	867,430,256	865,607,968	530,991,693	527,313,467	945,052,693	928,527,715	533,213,399	531,396,360
	br_immed_spec	162,934,076,942	161,823,926,703	57,001,296,547	57,368,768,278	166,204,140,982	166,148,106,331	53,731,544,162	53,413,102,849	187,570,979,531	187,369,922,999	56,148,023,018	55,458,485,945
	br_indirect_spec	6,721,525,313	6,611,471,069	7,115,777,950	7,153,833,402	6,655,249,425	6,711,447,560	7,403,454,222	7,356,026,734	6,787,157,764	6,845,201,849	7,671,416,657	7,574,371,802
	br_return_spec	4,240,896,555	4,165,772,305	4,674,275,366	4,704,811,432	4,221,222,622	4,277,675,115	4,953,923,562	4,916,041,660	4,338,760,739	4,390,467,287	5,207,219,742	5,153,005,116

Appendix F

Miler Rabin

Table F.1: General Benchmarks for Miller Rabin - Software Optimizations

Benchmarks	**Method 1**		**Method 2**				**Multithreading Method 2**			
	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
context-switches	58,495	75,428	444	916	7,572	106,675	12,532	25,011	N/A	N/A
page-faults	111	130	111	111	110	110	10,507	20,897	N/A	N/A
cycles	1,234,415,357,026	798,578,478,024	24,387,656,932	48,887,382,863	488,823,679,356	4,895,647,726,966	1,827,572,711	4,522,205,566	N/A	N/A
instructions	1,188,461,916,165	768,053,201,869	22,979,713,807	46,088,433,723	461,226,183,903	4,620,820,476,436	1,745,612,559	4,651,210,683	N/A	N/A
seconds	690.973606098	174.872790763	13.648183706	27.381384148	273.709273963	2739.736521662	5.192182217	10.410437517	N/A	N/A
cache-misses	43,195,907	143,543,462	1,748,512	3,320,065	33,438,989	344,484,591	3,766,812	5,510,127	N/A	N/A
mem_access	704,421,235,236	449,798,394,925	13,753,289,642	27,572,563,770	275,575,654,325	2,760,197,604,234	1,342,009,809	2,782,432,076	N/A	N/A
mem_access_rd	485,799,074,899	308,509,758,524	9,512,144,400	19,070,146,411	190,566,423,436	1,908,013,244,663	713,792,838	1,649,074,326	N/A	N/A
mem_access_wr	218,613,758,931	141,636,172,485	4,247,573,810	8,515,632,490	85,087,962,821	852,163,722,306	348,445,702	823,238,695	N/A	N/A

Table F.2: General Benchmarks for Miller Rabin - Compiler Optimizations with Software Optimizations

Benchmarks	**Method 1**				**Method 2**								**Multithreading Method 2**							
	O2		**O3**		**O2**				**O3**				**O2**				**O3**			
	Normal	Multithreading	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
context-switches	24,592	81,602	26,168	95,716	12,587	360	N/A	N/A	207	324	N/A	N/A	10,858	22,011	N/A	N/A	10,218	22,345	N/A	N/A
page-faults	111	131	112	130	110	110	N/A	N/A	111	110	N/A	N/A	10,506	20,898	N/A	N/A	10,508	20,898	N/A	N/A
cycles	523,179,582,500	543,243,421,289	499,538,644,477	546,613,835,428	15,746,551,258	30,047,964,202	N/A	N/A	14,946,032,415	29,748,792,141	N/A	N/A	2,334,744,801	3,202,165,980	N/A	N/A	1,256,701,944	2,703,776,247	N/A	N/A
instructions	319,214,161,200	336,513,693,599	305,306,947,984	340,954,318,137	8,782,226,157	16,774,062,352	N/A	N/A	8,355,251,956	16,619,694,457	N/A	N/A	1,368,194,543	2,121,814,945	N/A	N/A	757,606,621	1,668,432,327	N/A	N/A
seconds	292.891665947	152.706866776	279.755920462	102.251458860	9.477200115	16.815506940	N/A	N/A	8.388323421	16.651904282	N/A	N/A	3.867477558	7.742098412	N/A	N/A	3.941819254	7.762215084	N/A	N/A
cache-misses	18,573,176	136,223,846	17,733,169	184,087,384	5,479,785	2,719,841	N/A	N/A	1,195,771	2,654,593	N/A	N/A	2,849,788	5,891,159	N/A	N/A	1,940,734	4,527,300	N/A	N/A
mem_access	13,500,726,299	20,012,514,519	12,886,807,200	21,643,334,377	304,311,694	209,647,146	N/A	N/A	100,750,290	211,525,429	N/A	N/A	150,366,029	282,432,077	N/A	N/A	106,463,880	229,255,753	N/A	N/A
mem_access_rd	8,096,262,286	11,191,841,533	7,741,230,029	11,998,997,077	179,133,460	107,953,558	N/A	N/A	53,172,324	108,365,969	N/A	N/A	50,825,428	95,563,892	N/A	N/A	42,232,462	116,402,434	N/A	N/A
mem_access_wr	5,405,649,555	8,747,167,979	5,127,601,561	9,640,021,754	154,846,123	100,770,887	N/A	N/A	48,972,073	102,987,010	N/A	N/A	50,203,552	93,328,921	N/A	N/A	40,122,711	116,734,170	N/A	N/A

Table F.3: L1 Cache Benchmarks for Miller Rabin - Software Optimizations

Benchmarks	**Method 1**		**Method 2**				**Multithreading Method 2**			
	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
L1-dcache-loads	704,400,891,841	450,878,012,091	13,756,514,268	27,582,430,500	275,604,624,798	2,760,221,820,150	1,935,685,747	2,921,968,108	N/A	N/A
l1d_cache_refill	42,812,866	143,944,398	1,753,967	3,257,598	33,373,916	342,236,848	3,616,081	5,298,176	N/A	N/A
l1d_cache_rd	485,734,174,547	308,887,536,822	9,505,225,209	19,059,815,626	190,493,245,315	1,908,082,315,196	921,059,058	1,917,821,206	N/A	N/A
l1d_cache_wr	218,608,505,382	141,760,544,954	4,247,863,133	8,513,658,858	85,079,180,284	852,188,349,369	315,079,159	882,470,851	N/A	N/A
l1d_cache_wb	5,491,311	55,984,070	450,283	660,622	8,385,753	73,620,212	1,962,360	3,016,743	N/A	N/A
l1d_tlb_refill	20,451,520	93,978,580	1,452,867	2,782,039	27,122,039	274,981,765	1,250,227	2,088,330	N/A	N/A
l1d_cache_inval	149,841	53,126,012	154,081	318,698	3,257,439	32,131,178	79,936	227,251	N/A	N/A
L1-icache-loads	922,621,685,314	588,741,694,691	18,309,854,831	36,707,741,861	366,773,295,152	3,672,772,492,454	2,105,881,962	3,232,108,795	N/A	N/A
l1i_cache_refill	77,010,001	90,136,018	4,567,346	9,194,113	91,147,223	946,892,741	3,680,792	5,417,605	N/A	N/A
l1i_tlb_refill	1,296,747	7,500,516	257,832	502,872	4,782,747	51,582,850	134,063	219,976	N/A	N/A

Table F.4: L1 Cache Benchmarks for Miller Rabin - Compiler Optimizations with Software Optimizations

Benchmarks	Method 1				Method 2								Multithreading Method 2							
	O2		O3		O2				O3				O2				O3			
	Normal	Multithreading	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
L1-dcache-loads	13,504,936,737	20,013,939,627	12,862,396,501	21,561,262,925	323,429,729	205,220,800	N/A	N/A	107,962,048	204,179,245	N/A	N/A	120,958,512	286,091,403	N/A	N/A	153,263,079	278,134,474	N/A	N/A
ll d_cache_refill	18,612,310	135,529,715	17,878,597	183,761,238	5,607,449	2,657,981	N/A	N/A	1,208,947	2,625,170	N/A	N/A	2,595,063	6,668,958	N/A	N/A	3,131,879	5,623,390	N/A	N/A
ll d_cache_rd	8,109,070,126	11,222,194,538	7,739,684,977	11,986,546,163	184,735,993	108,546,001	N/A	N/A	54,294,437	110,240,270	N/A	N/A	66,155,575	90,531,944	N/A	N/A	42,347,310	96,762,130	N/A	N/A
ll d_cache_wr	5,407,588,502	8,744,449,694	5,128,397,243	9,603,428,602	151,063,698	100,032,166	N/A	N/A	51,396,628	100,005,074	N/A	N/A	61,280,155	86,901,007	N/A	N/A	54,384,393	80,621,857	N/A	N/A
ll d_cache_wb	3,108,641	52,108,987	2,228,658	71,930,326	1,393,381	683,675	N/A	N/A	336,677	798,242	N/A	N/A	1,408,728	3,736,626	N/A	N/A	1,635,307	3,103,266	N/A	N/A
lld_tlb_refill	8,754,252	66,901,719	8,635,905	79,683,016	3,488,837	2,467,545	N/A	N/A	1,258,250	2,422,306	N/A	N/A	954,044	2,600,061	N/A	N/A	1,048,972	2,023,534	N/A	N/A
ll d_cache_inval	55,931	49,446,792	107,462	67,146,987	494,921	250,918	N/A	N/A	124,563	281,917	N/A	N/A	119,437	140,865	N/A	N/A	63,551	158,243	N/A	N/A
L1-icache-loads	335,867,680,412	345,812,403,837	315,639,255,920	328,433,283,868	9,754,608,045	19,427,420,195	N/A	N/A	9,324,095,932	18,541,403,374	N/A	N/A	956,840,662	2,038,035,988	N/A	N/A	649,925,608	1,344,219,295	N/A	N/A
lli_cache_refill	32,652,999	75,961,509	33,476,564	87,081,448	14,595,601	6,889,128	N/A	N/A	3,456,003	6,875,394	N/A	N/A	2,540,033	5,378,060	N/A	N/A	1,990,097	4,256,907	N/A	N/A
lli_tlb_refill	575,543	5,626,069	506,903	6,494,920	461,614	350,017	N/A	N/A	193,050	407,871	N/A	N/A	110,173	262,588	N/A	N/A	65,942	173,036	N/A	N/A

Table F.5: L2 Cache Benchmarks for Miller Rabin - Software Optimizations

Benchmarks	**Method 1**		**Method 2**				**Multithreading Method 2**			
	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
l2d_cache	172,512,036	407,499,082	8,768,808	17,446,566	174,089,698	1,804,121,363	15,026,289	25,586,722	N/A	N/A
l2d_cache_refill	503,884	1,862,594	101,081	349,200	3,857,950	31,502,804	2,511,540	4,901,677	N/A	N/A
l2d_cache_rd	128,678,141	273,783,286	7,135,609	13,947,293	140,138,284	1,419,668,978	5,306,276	16,516,000	N/A	N/A
l2d_cache_wr	43,849,053	135,962,757	1,865,289	3,414,106	34,040,202	355,251,490	2,071,764	5,444,251	N/A	N/A
l2d_cache_wb	62,413	256,928	13,751	48,095	451,744	4,276,304	929,418	2,185,969	N/A	N/A
l2d_cache_inval	0	0	0	0	0	0	0	0	N/A	N/A

Table F.6: L2 Cache Benchmarks for Miller Rabin - Compiler Optimizations with Software Optimizations

Benchmarks	**Method 1**				**Method 2**								**Multithreading Method 2**							
	O2		**O3**		**O2**				**O3**				**O2**				**O3**			
	Normal	Multithreading	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
l2d_cache	73,811,616	368,212,774	73,144,080	475,963,615	28,304,853	13,795,036	N/A	N/A	6,519,618	13,931,882	N/A	N/A	11,697,717	25,232,543	N/A	N/A	8,995,104	22,430,695	N/A	N/A
l2d_cache_refill	242,751	1,747,561	362,547	1,890,600	2,915,789	221,871	N/A	N/A	167,674	220,285	N/A	N/A	2,381,305	4,898,604	N/A	N/A	1,723,980	4,181,885	N/A	N/A
l2d_cache_rd	55,161,425	243,451,764	53,296,134	309,632,644	23,260,117	11,212,579	N/A	N/A	5,136,253	11,018,946	N/A	N/A	7,243,050	13,378,162	N/A	N/A	7,602,049	13,087,624	N/A	N/A
l2d_cache_wr	18,887,043	122,254,374	18,527,311	165,111,998	5,667,010	2,873,953	N/A	N/A	1,278,408	2,785,252	N/A	N/A	2,892,592	3,951,314	N/A	N/A	2,839,939	5,756,707	N/A	N/A
l2d_cache_wb	32,691	237,169	49,731	262,564	316,448	33,947	N/A	N/A	17,903	28,237	N/A	N/A	1,157,676	1,864,215	N/A	N/A	663,437	1,676,650	N/A	N/A
l2d_cache_inval	0	0	0	0	0	0	N/A	N/A	0	0	N/A	N/A	0	0	N/A	N/A	0	0	N/A	N/A

Table F.7: Branch Predictions and Speculation Benchmarks for Miller Rabin - Software Optimizations

Benchmarks	**Method 1**		**Method 2**				**Multithreading Method 2**			
	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
br_pred	119,092,387,162	79,971,964,510	2,159,899,097	4,329,960,998	43,290,730,059	433,838,003,326	210,846,242	661,706,466	N/A	N/A
branch-misses	22,655,503,617	14,264,643,956	453,646,795	911,893,785	9,138,561,339	91,565,302,169	31,561,740	89,433,544	N/A	N/A
br_immed_spec	114,327,927,453	75,846,301,505	2,129,063,831	4,269,502,950	42,633,458,553	427,098,553,116	284,078,859	454,946,818	N/A	N/A
br_indirect_spec	4,766,921,930	4,023,537,222	33,315,686	67,611,324	667,935,416	6,770,577,487	14,839,008	24,003,265	N/A	N/A
br_return_spec	4,177,843,578	3,414,625,897	30,627,349	62,073,176	613,613,296	6,206,045,354	14,130,761	19,791,923	N/A	N/A

Table F.8: Branch Predictions and Speculation Benchmarks for Miller Rabin - Compiler Optimizations with Software Optimizations

Benchmarks	**Method 1**				**Method 2**								**Multithreading Method 2**							
	O2		**O3**		**O2**				**O3**				**O2**				**O3**			
	Normal	Multithreading	Normal	Multithreading	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums	5000 Nums	10,000 Nums	100,000 Nums	1,000,000 Nums
br_pred	80,135,841,537	84,038,400,531	76,636,388,703	85,110,806,528	2,196,271,385	4,187,899,602	N/A	N/A	2,088,395,405	4,155,425,105	N/A	N/A	319,533,673	552,967,080	N/A	N/A	252,182,651	448,612,038	N/A	N/A
branch-misses	17,955,338,681	18,048,707,009	17,191,123,611	18,098,370,199	509,330,461	1,040,541,570	N/A	N/A	517,084,710	1,030,932,033	N/A	N/A	60,119,120	101,237,441	N/A	N/A	38,914,238	73,671,443	N/A	N/A
br_immed_spec	77,748,996,109	81,077,538,494	74,363,986,924	81,816,217,422	2,143,561,496	4,172,922,541	N/A	N/A	2,084,594,897	4,145,872,675	N/A	N/A	192,208,142	609,299,294	N/A	N/A	194,327,130	450,462,076	N/A	N/A
br_indirect_spec	2,408,015,927	3,132,816,002	2,297,597,243	3,307,499,079	51,265,856	25,172,205	N/A	N/A	12,191,862	23,950,120	N/A	N/A	8,771,695	30,852,563	N/A	N/A	10,136,819	22,915,670	N/A	N/A
br_return_spec	2,007,795,372	2,568,481,934	1,912,352,958	2,696,994,072	36,352,279	19,731,983	N/A	N/A	9,446,308	19,013,366	N/A	N/A	6,848,379	23,419,969	N/A	N/A	6,666,114	18,269,878	N/A	N/A

Appendix G

Figures

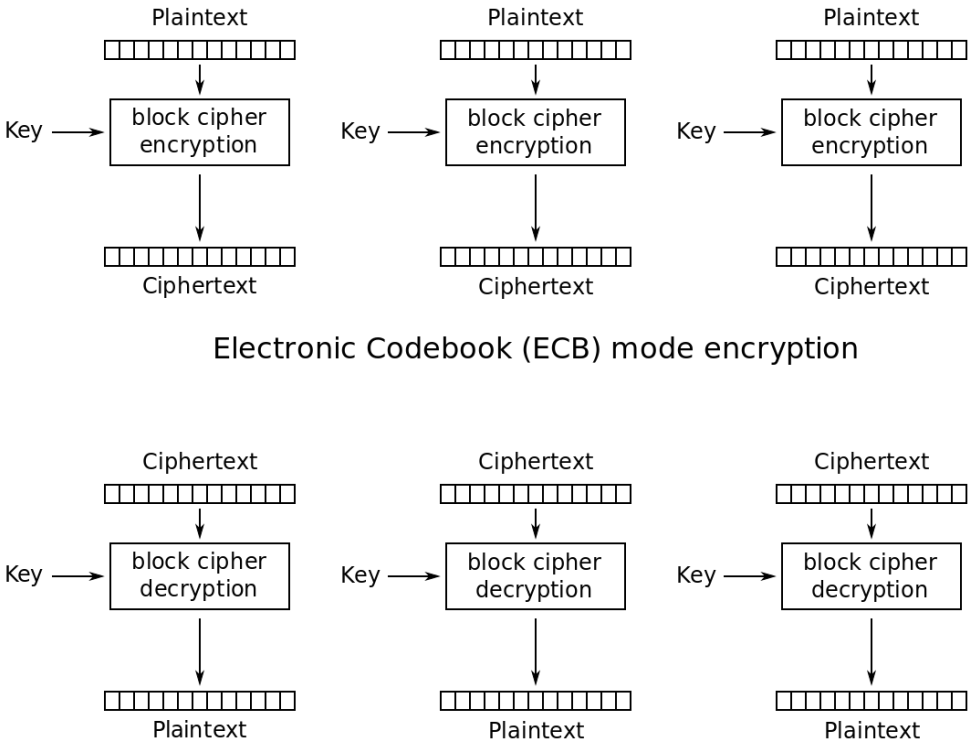
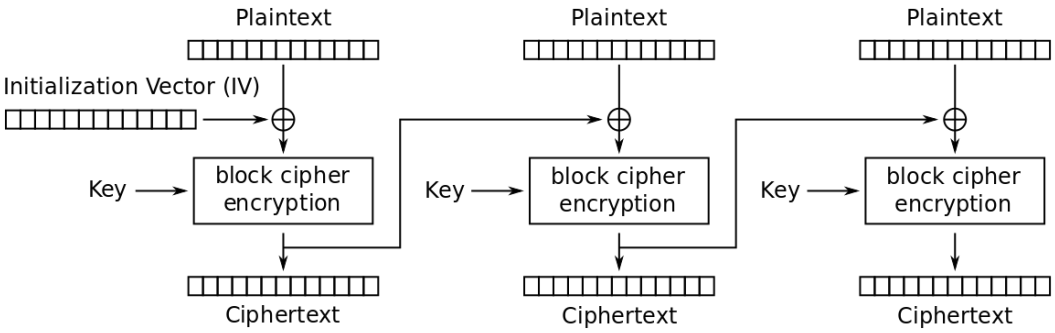
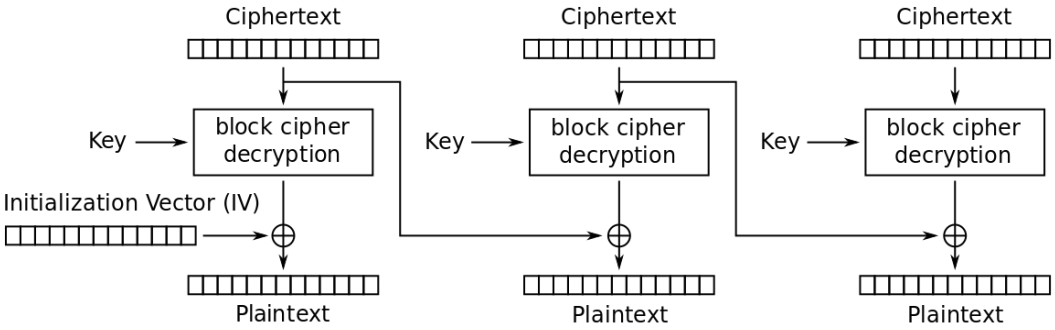


Figure G.1: Electronic Code Book (ECB) mode of operation [6]

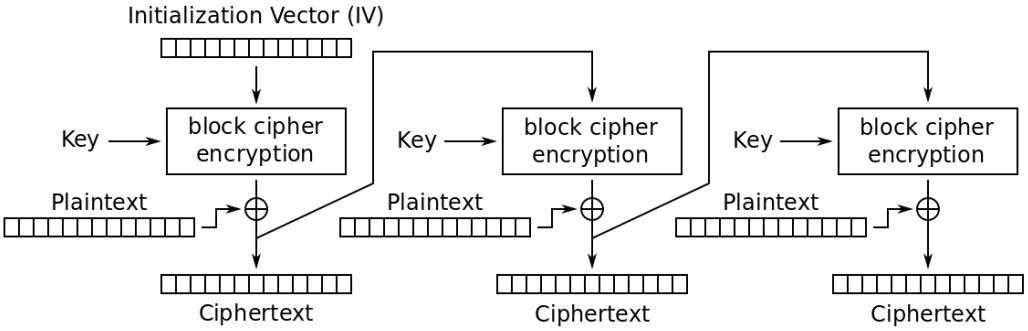


Cipher Block Chaining (CBC) mode encryption

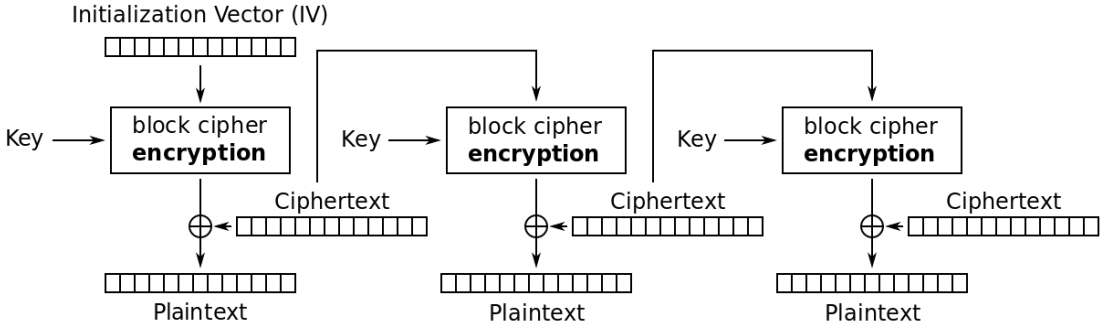


Cipher Block Chaining (CBC) mode decryption

Figure G.2: Cipher block chaining (CBC) mode of operation [6]



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

Figure G.3: Cipher feedback (CFB) mode of operation [6]