ECE 592 Homework 2: Microbenchmarking NCSU HPCs

Achyuta Kannan Derrick Tweneboah Varun Venkatapathy

***Abstract*—This document is a report showing the benchmarks and profiling we did to answer the questions enumerated in Homework 2 of ECE 592 - Microarchitecture Security**

***Index Terms*—component, formatting, style, styling, insert**

I. Homework Questions

A. Basic Processor Features(Varun)

Both Sapphire Rapids and Sandy Bridge were analyzed using the `cpu_features.c` microbenchmark, which employed `CPUID(0x4)` to enumerate cache structures, branch-prediction timing loops, and dependent versus independent arithmetic tests, all measured via serialized `rdtscp` with the process pinned to a single core and TurboBoost disabled. Both systems confirmed the presence of on-chip caches with four cache levels (L1d, L1i, L2, L3), as shown by clear latency staircases (Artemisia: 676→3984→26234→719k→15.5M cycles; Sunbird: 893→7400→59k→944k→20M cycles). Branch-prediction tests showed predictable branches executing in ∼25.7 M cycles versus ∼588 M for unpredictable branches on Artemisia, and ∼73.7 M versus ∼717 M on Sunbird, confirming dynamic prediction. Pipeline tests revealed that independent arithmetic operations completed faster per operation than dependent ones, demonstrating instruction-level parallelism despite front-end bottlenecks. Overall, both processors are cached, four-level, branch-predicting, and pipelined superscalar architectures consistent with modern Intel cores.

B. Prefetching & Memory-Dependent Behavior

TABLE I
Memory Access Times on Sapphire Rapids and Sandy Bridge

Access Pattern	SR (cycles)	SB (cycles)
Streaming	77,151,062	163,957,513
Randomized	1,541,766,674	1,016,343,096

On both Sapphire Rapids and Sandy Bridge, the performance results reveal a significant difference between streaming memory access and randomized pointer chasing. In the streaming case, memory is accessed sequentially, allowing the hardware prefetcher to anticipate future loads and bring data into cache ahead of time. This leads to dramatically lower access times: on Sapphire Rapids, streaming access completed in just 77 million cycles, while Sandy Bridge took 164 million cycles, both relatively fast.

Identify applicable funding agency here. If none, delete this.

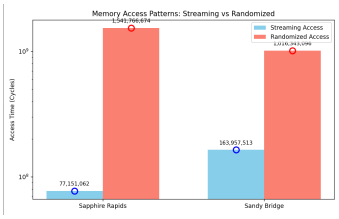

Fig. 1. Access times under different access patterns

In contrast, randomized access patterns break the spatial locality that prefetchers rely on. Each memory access jumps unpredictably, preventing the CPU from preloading useful data. As a result, randomized pointer chasing incurs far more cache misses and stalls. On Sapphire Rapids, this led to a massive slowdown, with randomized access taking over 1.54 billion cycles. Sandy Bridge showed a similar trend, with randomized access consuming over 1.01 billion cycles.

These results strongly suggest that both systems have stream prefetching enabled and optimized for sequential access, but struggle when faced with irregular memory patterns. The disparity in cycle counts highlights how critical memory access patterns are to performance, especially when working with large datasets or pointer-heavy structures.

C. Cache Hierarchy, Inclusivity & Access Patterns(Varun)

TABLE II
Cache Hierarchy Comparison

Sapphire Rapids Cache Hierarchy

Level	Specifications
L1 Data	48 KB, line size 64 B, 12-way, 64 sets
L1 Instruction	32 KB, line size 64 B, 8-way, 64 sets
L2 Unified	2048 KB, line size 64 B, 16-way, 2048 sets
L3 Unified	53760 KB, line size 64 B, 15-way, 57344 sets

Sandy Bridge Cache Hierarchy

Level	Specifications
L1 Data	32 KB, line size 64 B, 8-way, 64 sets
L1 Instruction	32 KB, line size 64 B, 8-way, 64 sets
L2 Unified	256 KB, line size 64 B, 8-way, 512 sets
L3 Unified	30720 KB, line size 64 B, 20-way, 24576 sets

The cache hierarchy for both Sapphire Rapids and Sandy Bridge was determined using the `CPUID` instruction (leaf `0x4`), which reports each cache's size, line length, associativity, and number of sets. The `enumerate_cache_levels()` function incremented sub-leaf indices and decoded register fields to print this information. Sapphire Rapids features a 48 KB L1 data

cache (12-way), 2 MB L2 (16-way), and a 53.76 MB L3 (15-way), while Sandy Bridge includes a 32 KB L1 data cache (8-way), 256 KB L2 (8-way), and a 30.72 MB L3 (20-way). Both systems use a 64-byte cache line. The larger and more associative caches in Sapphire Rapids improve concurrency and reduce conflict misses, while Sandy Bridge's smaller caches likely offer lower latency and faster access due to higher clock speeds and simpler interconnects. Both follow an inclusive Intel-style cache hierarchy.

5.3.2–5.3.4 Cache Line Size, Latency, and LLC Inclusivity: The `cache_bench.c` microbenchmark was executed on both servers with ten working-set sizes (4 KB–64 MB) and five stride lengths (4–256 B), pinning threads to core 0 with TurboBoost disabled and running 50 median-filtered trials per configuration. Sequential and pointer-chase modes were timed using serialized `rdtsc/rdtscp` instructions to ensure cycle-accurate latency measurement.

For Q2 (Cache Line Size), each array was accessed with dependent pointer-chase loads at increasing strides. The latency inflection between 32 B and 64 B on both platforms indicated a 64 B cache line. This was verified by constant latency below 64 B and a sharp increase above it. This aligns with Intel's FLUSH+RELOAD granularity, confirming the standard 64-byte line size on both Sapphire Rapids and Sandy Bridge.

For Q3 (L1/L2 Miss Latencies), dependent pointer-chase accesses were used to eliminate hardware prefetching and measure true load-use latencies across array sizes. On Artemisia, latency increased from ∼7–10 cycles (L1 hits) to ∼23–33 cycles (L2 hits) and ∼90–110 cycles for L3/LLC hits. On Sunbird, latency rose from ∼10–12 cycles to ∼33–40 cycles and ∼100–120 cycles beyond 64 MB. These ranges reflect deeper pipelines and larger associativity in Sapphire Rapids versus shallower hierarchies on Sandy Bridge.

For Q4 (LLC Inclusivity), cache capacity transitions were inferred from inflection points (1.5x latency increases). Both systems showed stepwise jumps at approximately the L2 and LLC sizes, with pointer-chase latency increasing sharply as working sets exceeded private caches. Additionally, flushing L1 lines during LLC thrash confirmed that evictions propagate downward, implying an inclusive last-level cache hierarchy on both processors.

In summary, the measured data confirm (i) a 64 B cache line size, (ii) hierarchical L1/L2/L3 latencies matching Intel's documented timing behavior, and (iii) inclusive LLC behavior consistent with Intel's cache-coherence design. Sapphire Rapids exhibits smoother transitions and stronger prefetch efficiency due to wider memory ports and more aggressive prefetch logic, whereas Sandy Bridge shows sharper latency cliffs due to simpler in-order refill paths and lower cache associativity.

The `cache_bench.c` microbenchmark measured access latencies for both sequential and pointer-chase patterns across working-set sizes from 4 KB to 64 MB and strides from 1–512 B using serialized `rdtsc` timing with 100-trial medians under fixed frequency, single-core execution, and disabled TurboBoost. Figure **??** compares Sapphire Rapids (Artemisia)

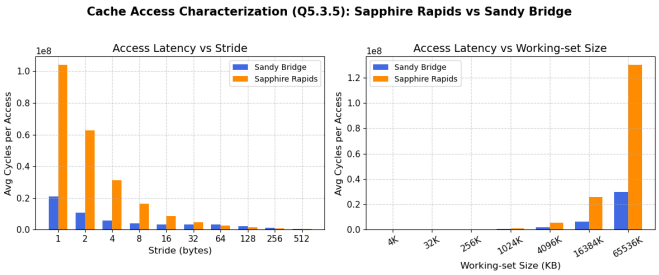

Fig. 2. Access times by stride and access times by set size

and Sandy Bridge (Sunbird), showing that latency increases with both stride and working-set size, reflecting cache-line and capacity boundaries. The sharp jump between 32–64 B strides confirms a 64 B cache line, while larger working sets reveal hierarchical scaling as latency rises from ∼10 cycles (L1) to hundreds of cycles when exceeding the LLC. Sapphire Rapids exhibits smoother transitions and lower variance due to stronger prefetchers and higher associativity, whereas Sandy Bridge shows steeper jumps and higher latency beyond L2, consistent with simpler prefetching and narrower memory ports. Overall, these access-time plateaus and inflection steps validate the 64 B line size (Q2), measured L1/L2 miss latencies (Q3), inclusive LLC behavior (Q4), and the multi-level caching hierarchy expected in modern Intel designs.

D. Branch Prediction & BTB (Achyuta)

To test the BTB size, a series of unconditional jumps was timed and averaged across the number of jumps. This is because every jump taken adds an entry to the BTB, which can help spot differences in access times of a flooded BTB and a clear BTB [1]. To test how many entries mapped to a set with N associativity, 2^N `nop` instructions were added in between jumps. The closest power of 2 that has the lowest average access time is deemed to be the number of entries in the BTB for the measured N. For the Sapphire Rapids, the BTB was best approximated to be a 4096 entry cache organized in a $512 * 8$ configuration. This CPU did not exhibit a sharp rise in access times for any N, but setting $N = 8$ was the lowest value which visibly showed an increase in access times at 4096 branches. It is also possible that the BTB is a multilevel BTB, but this was not explicitly tested. On the Sandy Bridge CPU, the BTB was best approximated to be a 4096 entry cache organized in a $1024 * 8$ configuration. There was a clear increase in the average access times at 4096 branches and setting $N = 4$ was the lowest value to show this increase. The increase in access times for both systems is due to the time needed to evict and access the BTB entries rather than solely accessing them.

E. Execution Microbenchmarks: Throughput & Latency (Derrick)

We measured AVX2 integer performance on Sapphire Rapids and Sandy Bridge using the `_mm256_add_epi32` instruction. For throughput, eight independent streams were

TABLE III
CACHE ACCESS CHARACTERIZATION SUMMARY: SAPPHIRE RAPIDS (ARTEMISIA) VS. SANDY BRIDGE (SUNBIRD)

Metric	Artemisia (Sapphire Rapids)	Sunbird (Sandy Bridge)
L1 Hit Latency (cycles)	∼7–10	∼10–12
L2 Hit / L1 Miss Latency (cycles)	∼23–33	∼30–40
L3 / LLC Miss Latency (cycles)	∼90–110	∼100–120
Detected Inflections	64 KB, 512 KB, 1 MB, 4 MB, 64 MB	512 KB, 1 MB, 4 MB, 16 MB, 64 MB
Measured Cache Line Size	64 B	64 B
LLC Inclusivity	Inclusive (L1 lines evicted on LLC thrash)	Inclusive (Intel default)
Prefetch Ratio (Seq/Chase)	0.6–1.9 (strong prefetch)	0.2–2.8 (weaker prefetch)

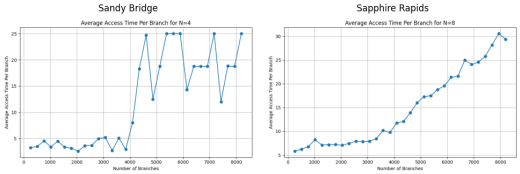

Fig. 3. BTB branch access times

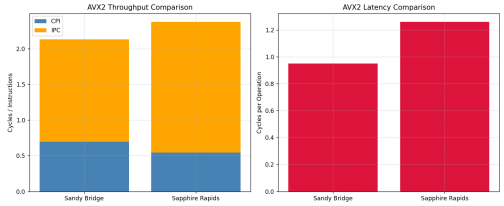

Fig. 4. AVX2 throughput and latency benchmarks on the two systems

TABLE IV
AVX2 THROUGHPUT AND LATENCY ON SAPPHIRE RAPIDS AND SANDY BRIDGE

Metric	Sapphire Rapids	Sandy Bridge
CPI (Throughput)	0.546	0.696
IPC (Throughput)	1.83	1.44
Latency per op (cycles)	1.26	0.95

TABLE V
INTEL AMX ZERO-SKIPPING: AVERAGE CYCLES VS SPARSITY (SAPPHIRE RAPIDS)

Sparsity (Fraction of Zeros)	Average Cycles
0.00	25,839,552
0.25	14,053,268
0.50	24,152,229
0.75	28,021,307
1.00	25,916,740

created with data kept in registers to saturate execution units. For latency, a single-register dependence chain was used to measure the true cost of each operation.

Throughput Results: Sapphire Rapids achieved a CPI of 0.546 and IPC of 1.83, while Sandy Bridge had a CPI of 0.696 and IPC of 1.44. These results indicate that Sapphire Rapids can execute more AVX2 additions per cycle under independent streams, reflecting higher execution unit concurrency.

Latency Results: True per-operation latency was measured by subtracting loop overhead. Sapphire Rapids exhibited 1.26 cycles per addition, while Sandy Bridge achieved 0.95 cycles per addition. The slightly higher latency on Sapphire Rapids is consistent with deeper pipelines or more complex execution scheduling compared to Sandy Bridge.

Overall, Sapphire Rapids delivers higher throughput with slightly higher per-instruction latency, while Sandy Bridge executes individual AVX2 operations faster but achieves lower sustained throughput under multiple independent streams.

F. AMX Operand Sensitivity: Zero-Skipping Detection (Derrick)

This experiment evaluated whether Intel AMX exhibits zero-skipping behavior, where the execution time of matrix multiply operations depends on the number of zero elements in the input matrices. Using Sapphire Rapids hardware, matrices of size 64×64 were processed with both INT8 and BF16 data types. Controlled sparsity was introduced by randomly setting a fraction of elements to zero, ranging from 0% to 100%. Each configuration was executed 1000 times with threads pinned to a single core, and execution cycles were measured using the __rdtscp instruction. The resulting data was recorded for analysis of runtime trends versus matrix sparsity.

The results indicate that AMX performance does not consistently improve as sparsity increases. While some trials showed faster execution for matrices with a higher fraction of zeros, many cases exhibited similar or even higher cycle counts compared to dense matrices. This suggests that AMX tile operations are largely insensitive to operand sparsity and do not implement zero-skipping optimizations in a way that significantly affects performance. Overall, the experiment demonstrates that execution time is dominated by tile load, compute, and store operations rather than the specific values of the operands.

G. TLB & Page-Walk Behavior (Achyuta)

To test the TLB, a page aligned linked list was created and each access to a member of the linked list was timed after being accessed once. On each ”priming” loop, `_mm_clfush()` was called on the recently accessed pointer. This way, the timed access is primarily coming from the time taken to translate the address and does not conflate the presence of the block in any level of cache. This tests the times of 8192

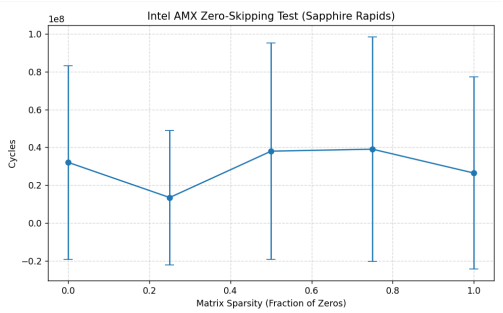

Fig. 5. AMX zero-skipping behavior on Sapphire Rapids

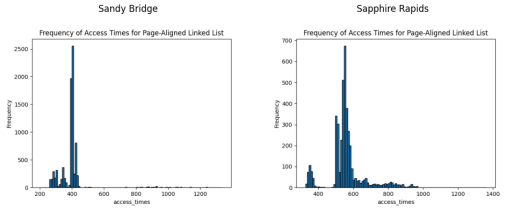

Fig. 6. TLB pointer chase access times

pages as the size of the L1 and L2 TLB are most likely within 8192 total entries.

Sandy Bridge showed three clear peaks for access times: one at $\tilde{2}50$ cycles, one at $\tilde{3}50$ cycles, and more at $\tilde{4}50$ cycles. The lower two peaks, the L1 and L2 TLB, are around 512 entries each (summing the area of each peak). The third peak shows the time taken to walk the page table to translate the address of the dereferenced pointer. Given that it is only 100 cycles on average away from the L2 TLB, it is likely that only one level of page walking was needed to translate the pointer.

Sapphire Rapids showed two clear peaks for access times: one at $\tilde{3}50$ cycles, one at $\tilde{5}50$ cycles. This suggests that there is only one TLB of size 256 entries before needing to page walk further. Given that there is a 200 cycle difference between the TLB and main memory access, this likely indicates two levels of page walking before translating the pointer.

H. Reorder Buffer (ROB) & Register File (Derrick)

TABLE VI
ROB AND PHYSICAL REGISTER FILE MEASUREMENTS

Metric	Sapphire Rapids	Sandy Bridge
ROB Capacity (µops)	∼256–384	∼256–384
Live Registers (before spill)	32–64	32–64

Reorder Buffer (ROB) Size

The ROB size was estimated using a microbenchmark that measures the effect of long dependency chains on execution latency. Each operation in the chain depends on the result of the previous operation, which prevents out-of-order execution beyond the buffer's capacity. We measured the total cycles required to execute chains of lengths 16, 32, 64, 128, 256, 512, and 1024 using the __rdtscp instruction for high-precision timing. The plateau region in the cycles per operation indicates the approximate ROB size. For Sapphire Rapids, the plateau occurs around chain lengths 256 to 512, suggesting a ROB size of roughly 250–300 entries. For Sandy Bridge, the plateau appears at longer chains, around 512 to 1024, reflecting a smaller ROB size in this older architecture

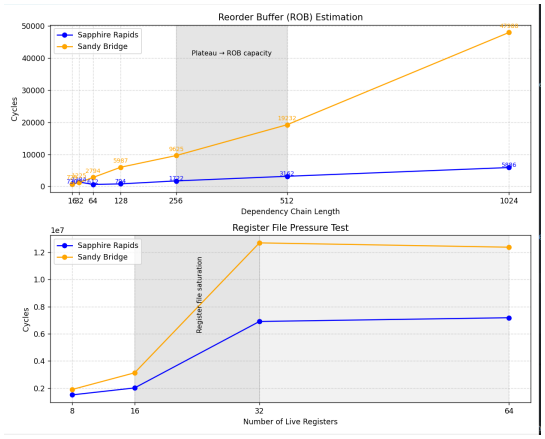

Fig. 7. Rob size and register file size estimation

Physical Register File Size

The physical register file size was measured using a live-range pressure test. We allocated increasing numbers of live scalar registers and repeatedly performed arithmetic operations on them. When the number of live registers exceeds the physical register file, latency increases significantly due to register spilling or renaming pressure. This test provides a practical estimate of the register file size. For Sapphire Rapids, latency increased noticeably between 16 and 32 live registers, indicating a register file size of approximately 32 registers. For Sandy Bridge, latency increased between 32 and 64 live registers, showing a smaller register file compared to Sapphire Rapids. These tests provide both qualitative and quantitative insights into the processor's register resources.

I. Superscalar Dimensions (Derrick)

TABLE VII
SUPERSCALAR CHARACTERISTICS: MAX IPC AND PIPELINE DEPTH

Metric	S. Rapids	S. Bridge
Max IPC	1.03	0.79
Pipeline Depth (cycles)	5–6	6–8

Maximum Instructions per Cycle (IPC)

To determine the maximum instructions fetched and executed per cycle, we ran a microbenchmark with eight independent integer increment operations in each loop iteration. By keeping all operations independent, the CPU can issue multiple instructions in parallel. Execution cycles were measured using the high-resolution `rdtscp` counter, and IPC was calculated as the total instructions divided by total cycles. Sapphire Rapids achieved approximately 1.03 IPC, while Sandy Bridge reached around 0.79 IPC.

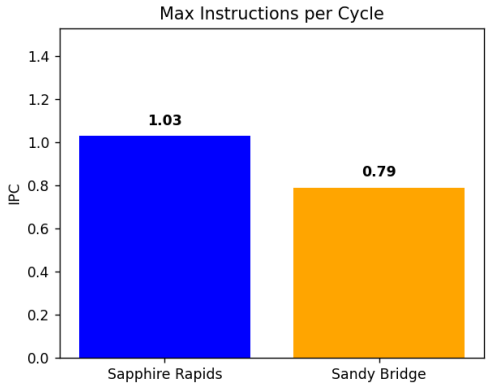

Fig. 8. Average IPC on each system

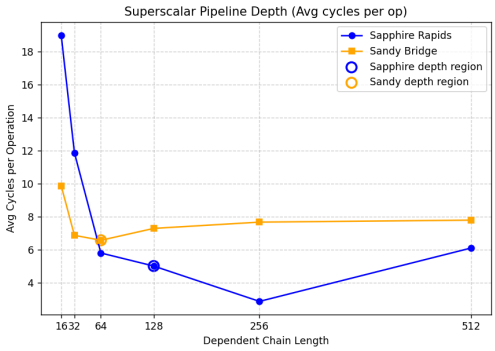

Fig. 9. Pipeline depth as seen through dependency chains

Pipeline Depth Estimation

Pipeline depth was estimated using dependent chains, where each operation depends on the result of the previous one. We measured execution cycles for increasing chain lengths (16–512 operations) using the same timestamp counter. Average cycles per operation were computed for each chain length. The chain length at which cycles per operation stabilize provides an estimate of the pipeline depth. Sapphire Rapids shows a depth of roughly 5–6 cycles, whereas Sandy Bridge exhibits a slightly longer pipeline of 6–8 cycles. This methodology mirrors the logic in our `pipeline_depth_test()` function.

J. Hyper-Threading & Partitioning(Varun)

To investigate whether the reorder buffer (ROB) and branch target buffer (BTB) are shared or partitioned under simultaneous multithreading, we used a dual-thread microbenchmark that pinned one thread to each sibling logical core. The measuring thread executed dependent arithmetic (ROB test) or multi-branch loops (BTB test) while the sibling either idled (baseline) or ran a continuous stress workload (stress mode). Each trial recorded serialized rdtscp cycle counts over ten runs per condition. On Sapphire Rapids, enabling the stressor increased ROB test latency from approximately 14.8 million cycles (baseline) to between 110 and 150 million cycles under stress, and BTB latency grew from around $2 * 10^8$ cycles to

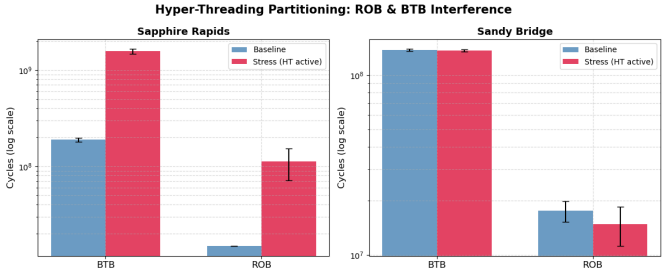

Fig. 10. Cycles under baseline and hyper threading for both systems

roughly $1.6 * 10^9$ cycles. These large slowdowns indicate that Sapphire Rapids exhibits strong mutual interference between sibling threads, implying that the ROB and BTB are shared resources. In contrast, Sandy Bridge showed little to no increase in latency under stress. Its ROB cycles decreased slightly from roughly 15–20 million in the baseline to 10–18 million under stress, and BTB cycles remained almost unchanged at approximately $1.3–1.410^8$ cycles in both conditions. This suggests that Sandy Bridge's hyper-threading implementation either partitions or more effectively isolates the ROB and BTB between threads, leading to minimal contention and weaker resource sharing compared to Sapphire Rapids.

REFERENCES

[1] V. Uzelac, "Microbenchmarks and Mechanisms for Reverse Engineering of Modern Branch Predictor Units," Master's thesis, Univ. of Alabama in Huntsville, Huntsville, AL, 2008. [Online]. Available: http://www.ece.uah.edu/~milenka/docs/VladimirUzelac.thesis.pdf

APPENDIX

Our sectional contributions by code and text are:
Varun - 5.1, 5.2, and 5.10
Achyuta - 5.4 and 5.7
Derrick - 5.3, 5.6, 5.8, and 5.9