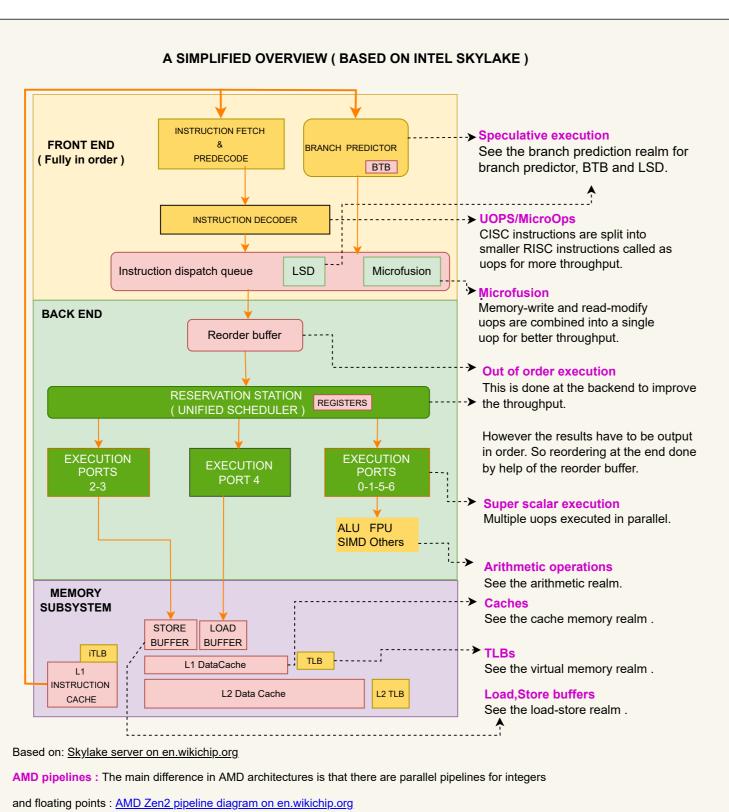
X86 CPUs & Performance

Die photo of a quadcore CPU , Copyright of Intel

LAST UPDATE DATE: 03 JAN 2023 FOR LATEST VERSION: www.github.com/akhin/microarchitecture-cheatsheet AUTHOR: AKIN OCAL akin_ocal@hotmail.com



INSIDE INDIVIDUAL CORE



PIPELINE PARALLELISM & PERFORMANCE Pipeline diagrams: The diagrams below in the following topics are outputs from an online microarchitecture analysis P Predecoded tool UICA and they represent parallel execution through cycles. Q Added to IDQ I Issued Rows are multiple instructions being executed at the same time. Ready for dispatch Columns display how instruction state changes through cycles. IPC: As for pipeline performance, typically IPC is used. It stands for "instructions per cyle". A higher IPC value usually means a better throughput. You can measure IPC with perf : https://perf.wiki.kernel.org/index.php/Tutorial Instruction lifecycle states Rate of retired instructions: Apart from IPC, number of retired instructions should be checked. Retired instructions in UICA diagrams are not committed/finalised as they were wrongly speculated. On the other hand executed instructions are the ones which were finalised. Therefore a high rate of retired instructions indicates low branch prediction rate. **CONTENTION FOR EXECUTION PORTS IN THE PIPELINE** Possible Ports | Actual Port | 0 In the example above, all instructions are working on different registers, but SHR, ADD, DEC instructions are competing for ports 0 and 6. SHR and DEC are getting executed after ADD instruction. Also notice that there is longer time between E(executed) and R(retired) states of instruction ADD as retirement has to be done in-order whereas execution is out-of-order. Reference: Denis Bakhvalov's article INSTRUCTION STALLS DUE TO DATA DEPENDENCY In the example above, there are 2 dependency chains, each marked with a different colour. In the first red coloured one, 2 instructions are competing for RAX register and notice that the second instruction gets executed after the first one. And the same applies to the 2nd purple pair. Reference: Denis Bakhvalov's article

RDTSCP INSTRUCTION FOR MEASUREMENTS TSC (time stamp counter) is a special register that counts CPU cycles. RDTSCP can be used to read the TSC value which then can be used for measurements.. It can also avoid out-of-order execution effects to a degree : it does wait until all previous instructions have executed and all previous loads are globally visible. (From Intel Software Developer's Manual Volume2 4.3, April 2022) Intel's How to benchmark code execution times whitepaper has details of using RDTSCP instruction. AMD Programmers Manual Vol3 states: RDTSCP forces all older instructions to retire before reading the timestamp counter **ESTIMATING INSTRUCTION LATENCIES** You can use Agner Fog's Instruction tables to find out instructions' reciprocal throughputs (clock cycle per instruction). As an example, reciprocal throughput of instruction RDTSCP is 32 on Skylake microarchitecture: -> 1 cycle @4.5GHZ (highest frequency on Skylake) is 0.22 nanoseconds -> 32*0.22=7.04 nanoseconds So its resolution estimation is about 7 nanoseconds on a 4.5 GHz Skylake CPU. You have to recalculate it for different microarchitectures and clock speeds. HYPERTHREADING / SIMULTANEOUS MULTITHREADING Hyperthreading name is used by Intel and it is called as "Simultaneous multithreading" by AMD. In both resources including caches and execution units are shared. Reference : Agner Fog`s microarchitecture book has "multithreading" sections for each of Intel and AMD microarchitectures Regarding using it, if your app is data-intensive, halved caches won't help. Therefore it can be disabled it via BIOS settings. In general, it moves the control of resources from software to hardware and that is usually not desired for performance critical applications. **DYNAMIC FREQUENCIES** Modern CPUs employ dynamic frequency scaling which Max level ← means there is a min and a max frequency per CPU C0 - Normal execution ← → Pi ACPI : ACPI defines multiple power states and modern CPUs implement those. P-State's are for performance and C states are for energy efficiency. In order to switch Intel has various tunability options and the most well to Pstates, C-state known is TurboBoost. On AMD side there is Turbocore has to be brought

Number of active cores & SIMD AVX2/512 on Intel CPUs: Intel's power management policies are complex.

See the arithmetic and the multicore realms as number of active cores and some of AVX2/512 extensions also

Varying max clock speeds on AMD CPUs: Some AMD CPUs' cores have slightly varying max frequencies.

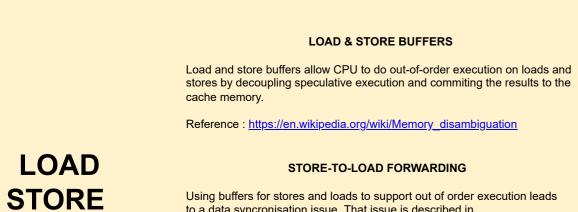
AVX : Up to 256 bits

AVX2 : Up to 256 bits

below are for 128 bit operations:

__m128 , 4 x 32 bit floating points

AVX512 : Up to 512 bits



Using buffers for stores and loads to support out of order execution leads to a data syncronisation issue. That issue is described in en.wikipedia.org/wiki/Memory_disambiguation#Store_to_load_forwarding As a solution, CPU can forward a memory store operation to a following load, if they are both operating on the same address. An example store and load sequence

mov [eax],ecx; STORE, Write the value of ECX register to the memory address which is stored in EAX register mov ecx,[eax]; LOAD, Read the value from that memory address ; (which was just used) and write it to ECX register

fetching a set of instructions in advance.

How: There are auxilliary hardware buffers.

Branch target buffer stores target addresses (instruction

be flushing the pipeline.

STORE-TO-LOAD FORWARDING & LHS & PERFORMANCE Based on Intel Optimization Manual 3.6.4, store-to-load forwarding may improve combined latency of those 2 operations. The reason is not specified however it is potentially LHS (Load-Hit-Store) problem in which the penalty is a round trip to the cache memory: https://en.wikipedia.org/wiki/Load-Hit-Store There are several conditions for the forwarding to happen. In case of a STORE BUFFER LOAD BUFFER successful forwarding, the steps 2 and 3 (a roundtrip to the cache) will be bypassed

L1 CACHE

What would happen without forwarding?: In the past, game consoles PlayStation3 and Xbox360 had PowerPC based processors which used inorder-execution rather than out-of-order execution. Therefore developers had to separately handle LHS by using restrict keyword and other methods : Elan Ruskin`s article

The conditions for a successful forwarding and latency penalties in case of

no-forwarding can be found in Agner Fog's microarchitecture book.

ARITHMETIC REALM ARITHMETIC INSTRUCTION LATENCIES You can see a set of arithmetic opertions from fast to slow below.

nteger division: 24-90 clock cycles

CACHE

MEMORY

REALM

architecture on 64 bit registers

number. Used <u>bartaz.github.io/ieee754-visualization</u> as visualiser mantissa - 23 bits 8 bits A floating point's value is calculated as: ±mantissa × 2 exponent IEEE754 also defines denormal numbers. They are very small / near zero numbers. As floating points are approximations, float GetInverseOfDiff(float a, float b) The clock cycles are based on Agner Fog`s <u>Instruction tables</u> & Skylake denormal numbers are needed to avoid an undesired case of : a!=b but a-b=0 Bitwise operations , integer add/sub : 0.25 to 1 clock cycle
Floating point add : 3 clock cycles Without denormals the code to the right would invoke a divide-by-zero exception. Reference : Bruce Dawson's article

Based on Agner Fog`s microarchitecture book, Intel CPUs have a penalty for denormal numbers, for ex: 129 clock cycles on Skylake. They also can be turned off on Intel CPUs. As for AMD side, the recent Zen architecture CPUs seemingly don't have the same performance degradation.

return 1.0f / (a - b);

return 0.0f;

FLOATING POINTS

in the memory layout. Below you can see all bits of 1234.5678 FP

X86 EXTENSIONS x86 extensions are specialised instructions. They have various categories X86 uses IEEE 754 standard for floating points. A 32 bit floating point consists of 3 parts from <u>cryptography</u> to <u>neural network operations</u>. Intel Intrinsics Guide is a good page to explore those extensions SSE (Streaming SIMD Extensions) is one of the most important ones. SIMD stands for "single instruction multiple data". SIMD instructions use wider registers to execute more work in a single go: i1 i2 i3 i4 + + + + = = = =

You can use those to maximise the CPU usage.

may affect the frequency while in Turboboost.

Reference: AMD's GDC22 presentation page6

In the example above, an array 4 integers (i1 to i4) are added to another array of integers (j1 to j4). The result is also an array of sums (s1 to s4). In this example, 4 add operations are executed by a single instruction. They play key role in compilers' vectorisation optimisations: GCC auto vectorisation Apart from arithmetic operations, they can be utilised for string operations as well:

Daniel Lemire's SIMD based JSON parser : <a href="https://github.com/simdjson/simdjs

Load & Store

L1 Data L1 Instruction

Cache Cache

m128d, 2 x 64 bit doubles m128i , 4 x 32 bit ints int int int _m128l , 2 x 64 bit long longs long long

X86 EXTENSIONS: SIMD DETAILS

Recent AMD CPUs support AVX & AVX2. Only the latest Zen4 architecture supports

Float

Float

Float

Double

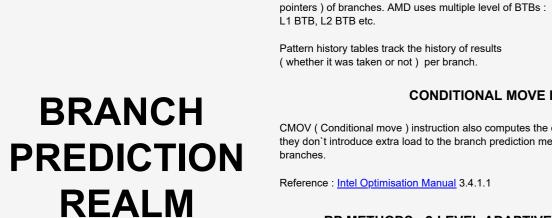
As for programming, there are also wider data types. The data type diagrams

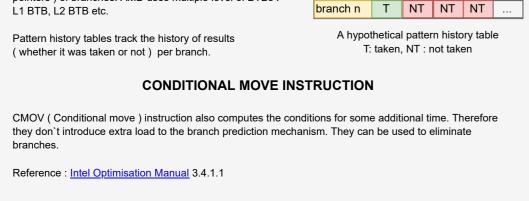
The most recent SIMD instruction sets for Intel CPUs are :

to C0 level

Note that as SIMD instructions require more power, therefore usage of some AVX2/512 extensions may introduce downclocking. They should be benchmarked. For details : Daniel Lemire`s article

REALM





BRANCH PREDICTION BASICS

Why: CPUs proactively fetch instructions of potentially upcoming branches to utilise the pipeline as much as

Gain if predicted correctly: If the right branch was predicted that will increase the throughput as it completed

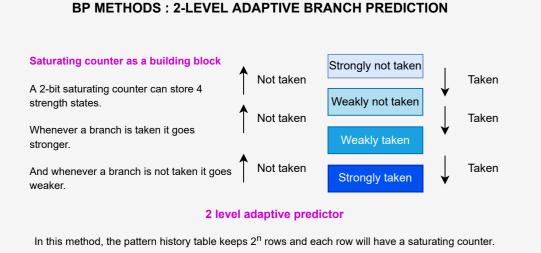
Penalty in case of misprediction: If the prediction was wrong, that prefetch will be a waste and the cost will

1 2 3 4 ...

T T NT T ...

branch ... NT NT NT T

What are branch instructions?: Unconditional ones (jmp), conditional ones (eg: jne), call/ret



A branch history register which has the history of last n occurences, will be used to choose which row will be used from the pattern history table. Reference: Agner Fog`s microarchitecture book 3.1.

BP METHODS: AMD PERCEPTRONS

A <u>perceptron</u> is basically the simplest form of machine learning. It can be considered as a linear array of Agner Fog mentions that they are good at predicting very long branches compared to 2-level adaptive branch prediction in his microarchitecture book 3.12. For details of perceptron based branch prediction: <u>Dynamic Branch Prediction with Perceptrons by Daniel</u> The output Y (in this case whether a branch Jimenez and Calvin Lin taken or not) is calculated by dot product of the weight vector and the input vector.

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)#Front-end

INTEL LSD (LOOP STREAM DETECTOR)

Intel LSD will detect a loop and stop fetching instructions to improve the frontend bandwidth. Several conditions mentioned in Intel Optimisation Manual 3.4.2.4 : • Loop body size up to 60 μops, with up to 15 taken branches, and up to 15 64-byte fetch lines. No CALL or RET. • No mismatched stack operations (e.g., more PUSH than POP). More than ~20 iterations. Note that LSD is disabled on Skylake Server CPUs. Reference :

DISABLING SPECULATIVE EXECUTION PATCHES You can consider disabling system patches for speculative execution related vulnerabilities such as Meltdown and Spectre for performance, if that is doable in your system. Those patches are not only microcode updates but they also need OS support. Kernel.org documentation : https://www.kernel.org/doc/html/latest/admin-guide/kernel- parameters.html

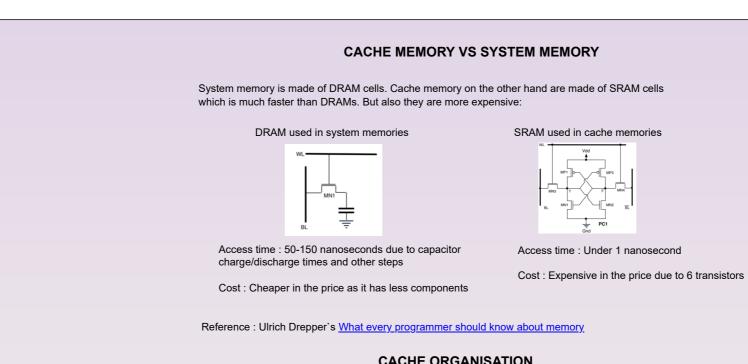
Red Hat Enterprise documentation: https://access.redhat.com/articles/3311301 Meltdown paper: https://meltdownattack.com/meltdown.pdf Spectre paper: https://spectreattack.com/spectre.pdf

As for max number of entries in BTBs, there are estimations made by stress testing the BTB with sequences of branch instructions : Intel Xeon Gold 6262 -> roughly 4K AMD EPYC 7713 -> roughly 3K

Reference: Marek Majkovski's article on Cloudflare blog

VIRTUAL MEMORY ADDRESS TRANSLATION

ESTIMATED LIMITS: HOW MANY IFS ARE TOO MANY?



CACHE ORGANISATION Caches are organised in multiple levels. As you go upper in that hierarchy, the capacity increases. Therefore LLC term used to indicate the last level of cache. 3 level data caches are currently the most common ones. Intel Broadwell architecture had 4 level caches in the past. Also upcoming AMD CPUs may come with 4 level of caches. LLC : Last level cache Cache line size is the unit of data transfer between the cache

can't predict. That is called as software prefetching.

and the system memory. It is typically 64 bytes. And the caches System memory are organised according to the cache line size. There is also instruction cache (iCache) which stores program instructions rather than data to improve throughput of CPU frontend. In case of a cache hit, the latency is typically single digit nanoseconds. And in case of a cache miss, we need a round trip to the system memory and total latency becomes 3 digit nanoseconds.

HARDWARE AND SOFTWARE PREFETCHING Hardware prefetchers detect patterns like streams Streaming (Ex: accessing to contiguous array members) and strides (Ex: accessing specific members in arrays of structs) and prefetch data and instruction to cache lines automatically. Developers can also use instruction _mm_prefetch to prefetch data explicitly for cases when hardware

Reference for image: It is taken from AMD's GDC22 presentation page44

N-WAY SET ASSOCIATIVITY Why: Cache capacities are much smaller than the system memory. Moreover, software can use various regions of their address space. So if there was one to one mapping of a fully sequential memory that would lead to cache misses most of the time. Therefore there is a need for efficient mapping between the cache memory and the system memory. How: In N-Way set associativity, caches are divided to groups of sets. And each set will have N cache blocks. The mapping information is stored in bits of addresses which has 3 parts: SET OFFSET used to determine used to determine the actual bytes identifier per cache block the set in a cache in the target cache block The pseudocode below shows steps for searching a single byte in the cache memory : Get tag, set and offset from the address

For each block in the current set (which we have just found out) if tag of the current block equals to tag (which we just have found out) read and return data using offset , it is a cache hit If there was no matching tag, it is a cache miss The level of associativity (the number of ways) is a trade off between the search time and the amount of system memory we can

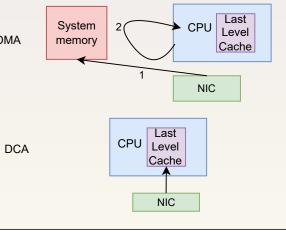
Temporal data is data that will be accessed in a short period of time. The term non-temporal data indicates that data will not be accessed any soon. (cold data). If the amount of non-temporal data is excessive in the cache, that is called as cache pollution. Non-temporal store instructions are introduced for this problem and they store data directly to the system memory by

BYPASSING THE CACHE: NON-TEMPORAL STORES & WRITE-COMBINING

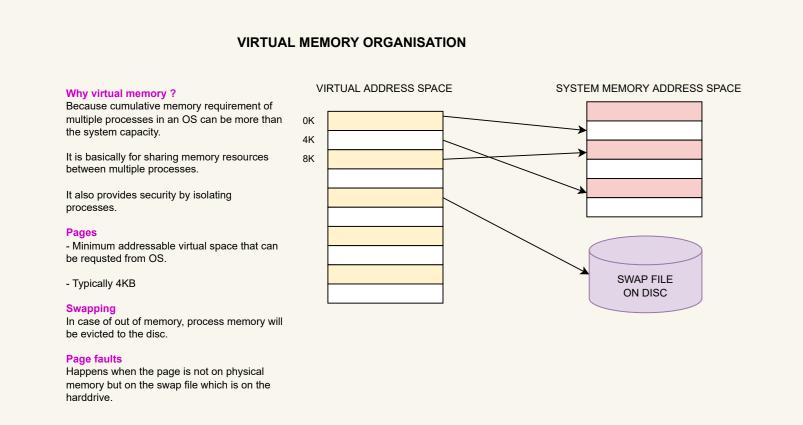
Write combining buffers are used with non-temporal stores. CPU will try to fill a whole cache line (typically 64byte) before committing to the system memory and only will send to the system memory when that buffer is filled. That is for reducing the load on the bus between the CPU and the system memory. DIRECT CACHE ACCESS

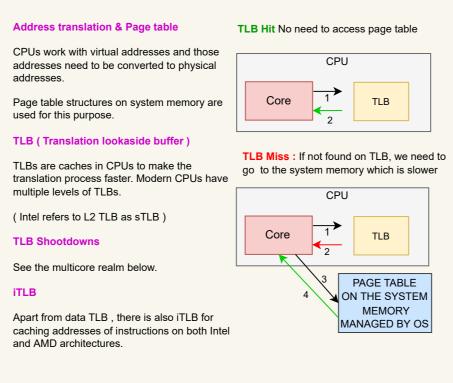
Modern NICs come with a DMA (Direct Memory Access) engine and can transfer data directly to drivers' ring buffers which reside on the system memory DMA mechanism doesn't require CPU involvement. Though mechanism initiated by CPU , therefore CPU support needed. DCA bypasses the system memory and can transfer to directly LLC of CPUs that support this feature.

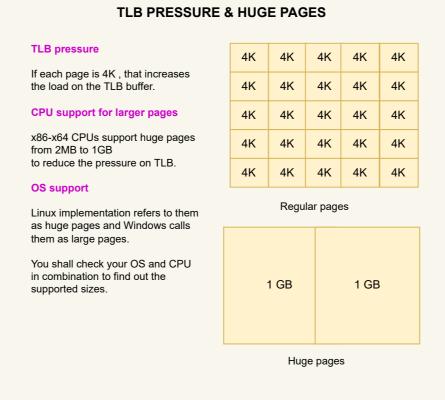
Intel refers to their technology as DDIO (Direct I/O). Reference : Intel documentation

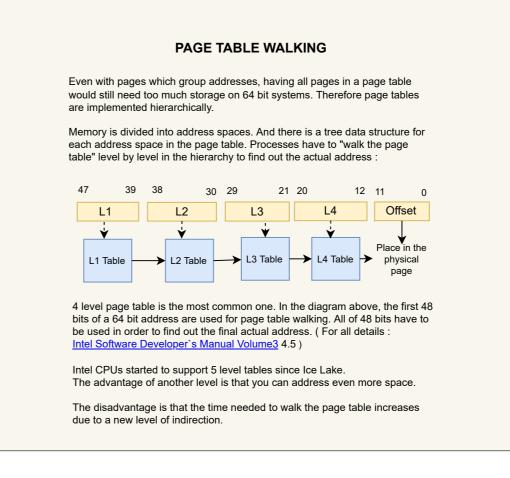


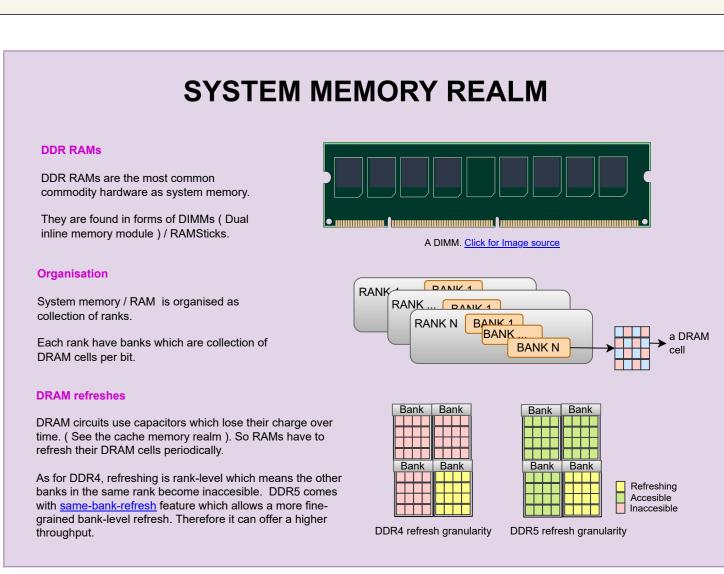
VIRTUAL MEMORY REALM











MULTICORE REALM

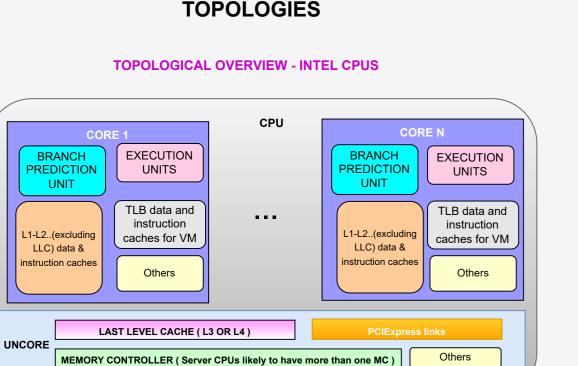


Diagram above aims to show resource per core and shared resources. Note that uncore in an Intelonly term to refer to CPU functionality which are not per core. **Exception of E-cores :** An exception to the above diagram is Intel's recent E-cores. E-cores are meant for power efficiency and paired with less resources. For ex: Alder Lake CPUs` E-cores also

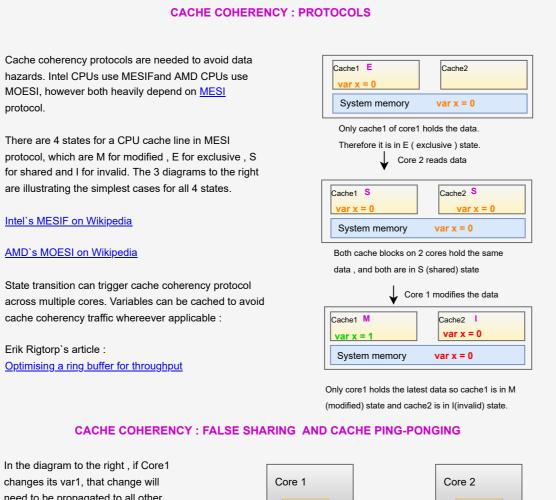
Reference: https://www.anandtech.com/show/16959/intel-innovation-alder-lake-november-4th

TOPOLOGICAL OVERVIEW - AMD CPUS

Most of AMD topology is similar to Intel. However starting from Zen microarchitecture, one key difference is CCXs. AMD CPUs are designed as group of 4 cores which is called as CCX (Core complex) , and there is one LLC per each CCX/quad core. Practically the maximum number of cores competing for the LLC (without simultanenous multithreading) is 4 in recent AMD CPUs. An example 8 core CPU with 2 CCXs: CPU CORE 1 CORE 2 | CORE 5 CORE 6 CORE 3 CORE 4 | CORE 7 CORE 8

Reference: https://en.wikichip.org/wiki/amd/microarchitectures/zen#CPU_Complex_.28CCX.29

COHERENCY



changes its var1, that change will need to be propagated to all other cores by the cache coherency protocol. That will lead to invalidation of cache areas associated with the shared cache line across all cores, even though it var1 var2 memory is used by only one core. That situation is called false sharing. Shared system memory cache line holding var1 for Core1 and var2 for Core2 If those happen in higher rates and if cache lines from system memory transferred between cores rapidly, that situation is called as cache ping-pong. **VIRTUAL MEMORY PAGE TABLE COHERENCY: TLB SHOOTDOWNS** Whenever a page table entry is modified by any of the cores, that particular TLB entry is invalidated in all cores via IPIs. This one is not done by hardware but initiated by operating system. IPI: Interprocessor interrupt, you can take "processor" as core in this context.

1. One of the cores modifies a table entry PAGE TABLE ON SYSTEM MEMORY

MEMORY REORDERINGS & SYNCRONISATION

MEMORY REORDERINGS The term memory ordering refers to the order in which the processor issues reads (loads) and writes (stores) . Based on Intel Software Developer's Manual Volume3 8.2.3.4 , there is only one kind of memory reordering that can happen. Loads can be reordered with earlier stores if they use different memory locations. That reordering will not happen if they use the same address CORE1 CORE2 ; x and y initially 0 ; x and y initially 0 mov [x], 1; STORE TO X mov [y], 1; STORE TO Y

mov [result1], y; LOAD FROM Y mov [result2], x; LOAD FROM X In case of reordering, result1 and result2 above can both end up as zero in both cores. Note that, apart from CPUs , also compilers can do memory reordering $\,:\!\underline{\mathsf{Jeff}}$ Preshing's article: Memory Ordering at Compile Time INSTRUCTIONS TO AVOID REORDERINGS

Reorderings can be avoided by using serialising instructions such as SFENCE, LFENCE, and MFENCE : $\underline{\text{Intel Software Developer's Manual Volume3}} \ \ \textbf{8.3 defines them as} :$ These instructions force the processor to complete all modifications to flags, registers, and memory by previous instructions and to drain all buffered writes to memory before the next instruction is fetched and There is also bus locking "LOCK" prefix (<a href="Intelligent Intelligent Int

which can be used as well to avoid reorderings.

ATOMIC OPERATIONS An atomic operation means that there will be no other operations going on during the execution. From point of execution, an atomic operation is indivisible and nothing can affect its execution The most common type of atomic operations are RMW (read-modify-write) operations. ATOMIC OPERATIONS & MEMORY ALIGNMENT

If an atomic instruction is used for a memory range which is split to multiple cache lines, that will lead to

locking the whole memory bus, instead of just the cache line.

Reference: Detecting and handling spit locks (in Linux kernel) on lwn.net ATOMIC RMW OPERATIONS: COMPARE-AND-SWAP CAS instruction (CMPXCHG) reads values of 2 operands. It then compares them and if they are equal, it swaps values. All the operations are atomic / uninterruptible. It can be used to implement lock free data structures.

ATOMIC RMW OPERATIONS: TEST-AND-SET Test-and-set is an atomic operation which writes to a target memory and returns its old value. It is typically used to implement spin locks. **PAUSE INSTRUCTIONS** Busy spinning applications (ex: user space spin locks) can degrade hyperthreading efficiency. There are

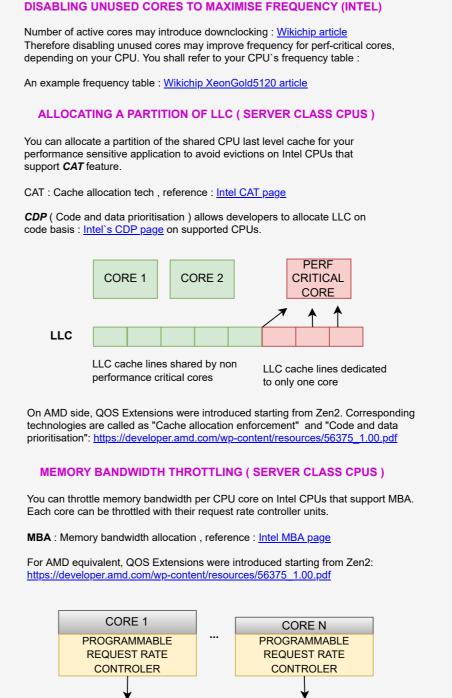
Note that the latency for PAUSE instruction on Skylake clients is an order of magnitude slower than other

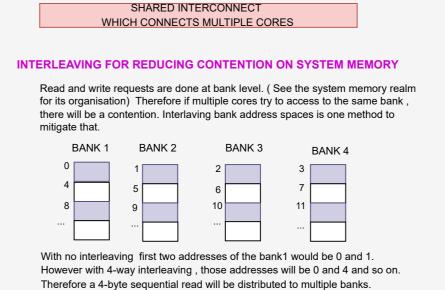
architectures: Intel Optimisation Manual 2.5.4 TRANSACTIONAL MEMORY Transactional memory areas are programmer specified critical sections. Reads and writes in those areas are done atomically. (<u>Intel Optimization Manual</u> section 16) However due to another hardware security issue, Intel disabled them from Skylake to Coffee Lake CPUs: https://www.theregister.com/2021/06/29/intel_tsx_disabled/

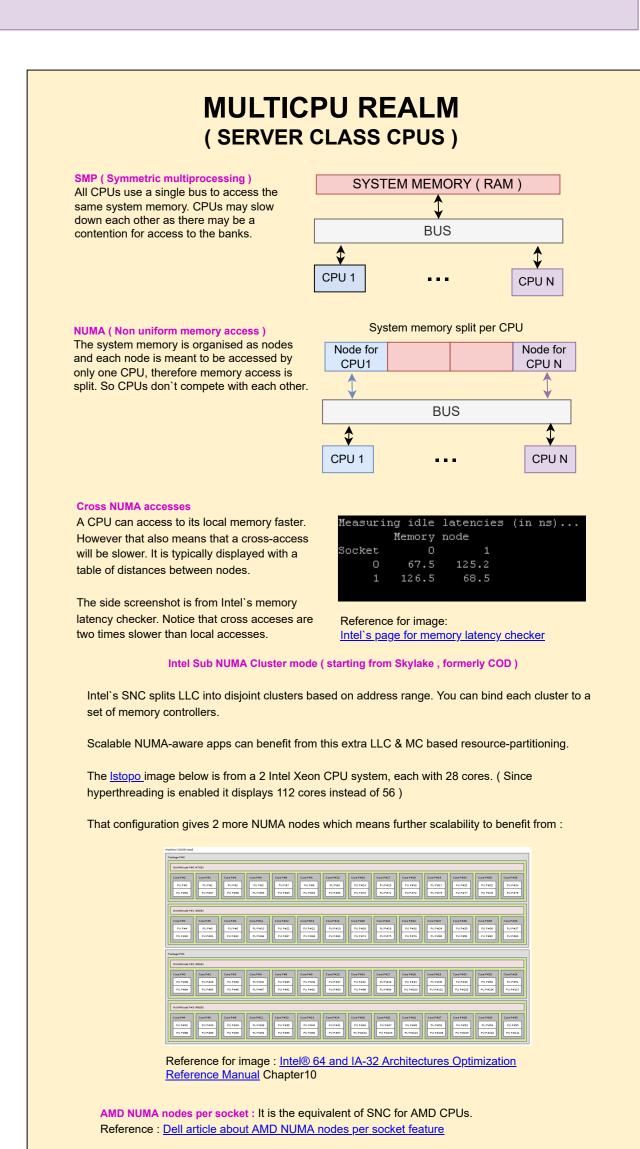
several pause instructions (PAUSE/ TPAUSE/UMWAIT/UMMONITOR) to help that.

AMD equivalent is called as "Advanced Syncronisation Facility". According to Wikipedia article, there are no AMD processors using it yet.

LIMITING CONTENTION BETWEEN CORES



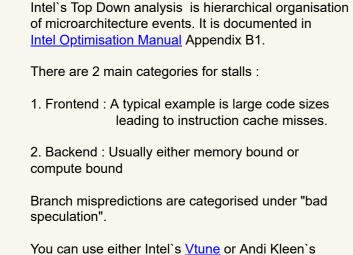




ACROSS

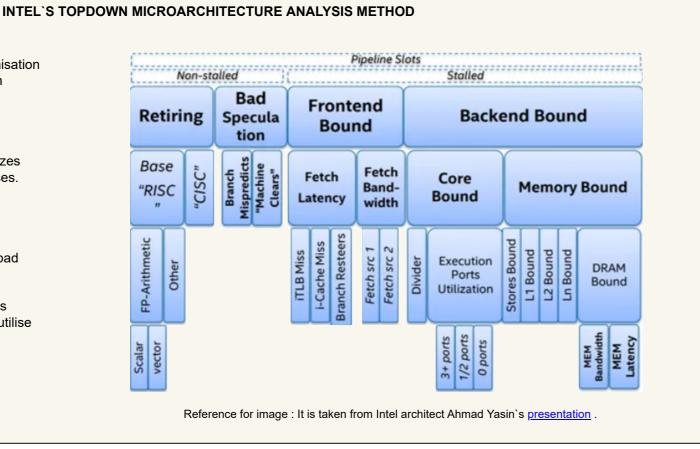
REALMS

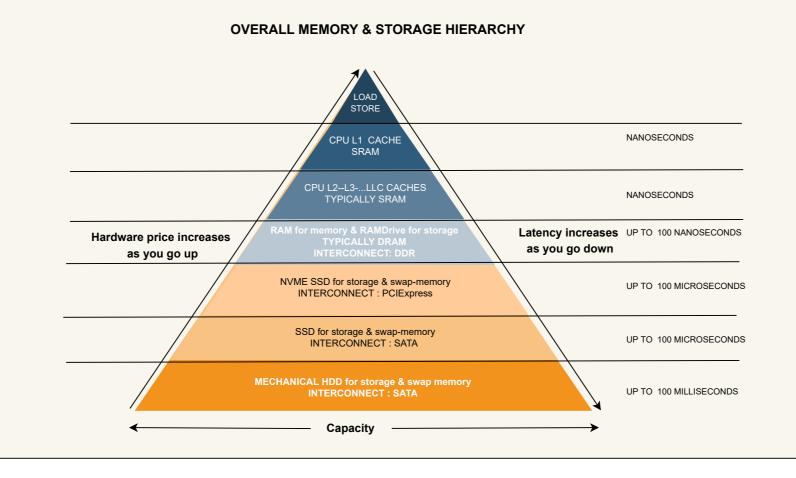
share L2 cache

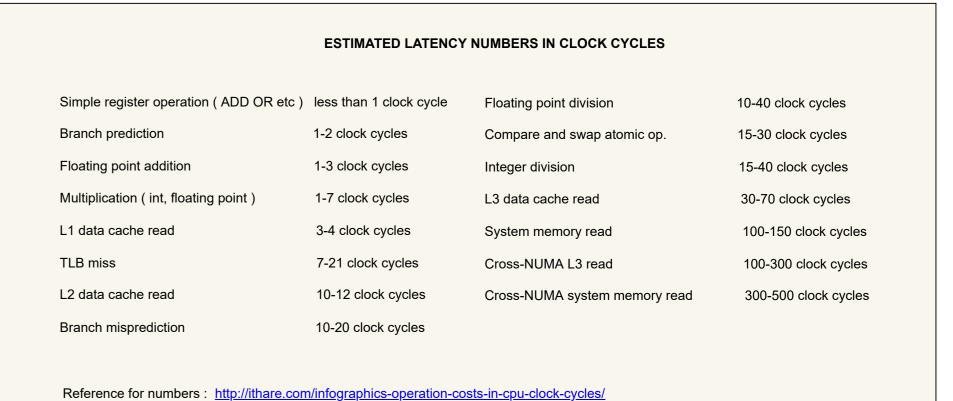


<u>Toplev</u> tool to make a top down analysis. Both utilise

Intel CPUs' performance monitoring counters.







Note: The reference is from 2016, however it still is good to show proportions between different events.