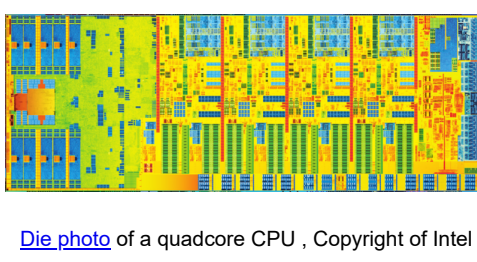


MICROARCHITECTURE CHEAT SHEET

X86 CPUs & Performance



Die photo of a quadcore CPU. Copyright of Intel

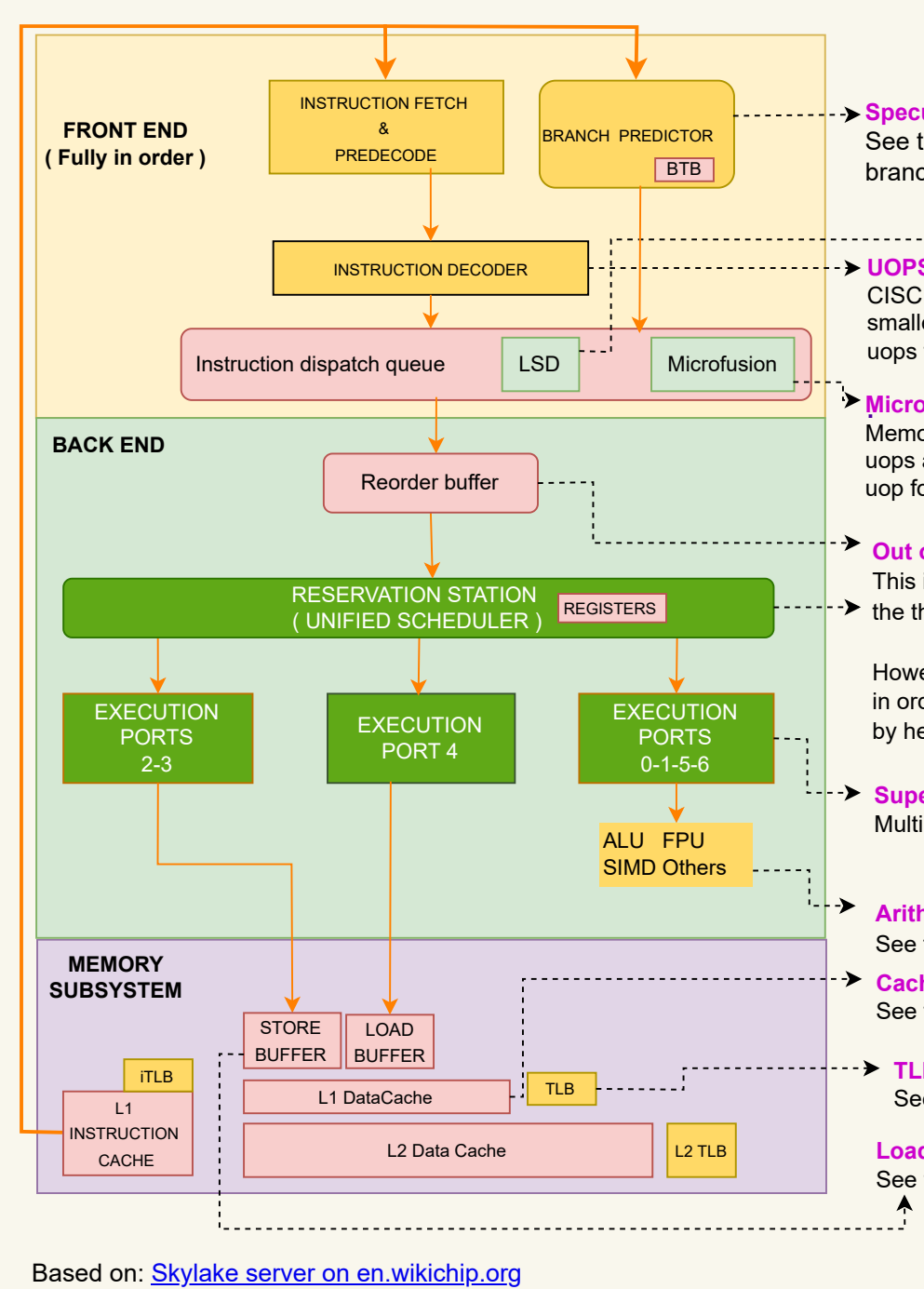
LAST UPDATE DATE : 18 OCT 2022

FOR LATEST VERSION: [www.github.com/ahm4/microarchitecture-cheatsheet](https://github.com/ahm4/microarchitecture-cheatsheet)

AUTHOR: AKIN OCAL akin_ocal@hotmail.com https://twitter.com/akin_ocal_dev

PIPELINE REALM : INSIDE AN INDIVIDUAL CORE

A SIMPLIFIED OVERVIEW (BASED ON INTEL SKYLAKE)



Based on: [Skylake server on en.wikichip.org](https://en.wikichip.org/wiki/skylake_server)

- Speculative execution**
See the branch prediction realm for branch predictor, BTB and LSD.
- MicroOps**
CISC instructions are split into smaller RISC instructions called as uops for more throughput.
- Microfusion**
Memory-write and read-modify ops are combined into a single uop for better throughput.
- Out of order execution**
This is done at the backend to improve the throughput.
However the results have to be output in order. So reordering at the end done by help of the ROB (Reorder buffer).
- Super scalar execution**
Multiple uops executed in parallel.
- Arithmetic operations**
See the arithmetic realm.
- Caches**
See the cache memory realm.
- TLBs**
See the virtual memory realm.
- Load/Store buffers**
See the load-store realm.

PIPELINE PARALLELISM & PERFORMANCE

Pipeline diagrams : The diagrams below in the following topics are outputs from an online microarchitecture analysis tool [UCCA](https://en.wikichip.org/wiki/analysis), and they represent parallel execution through cycles.

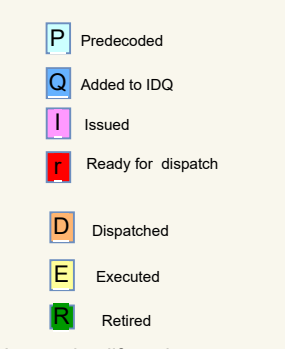
Rows are multiple instructions being executed at the same time.

Columns display how instruction state changes through cycles.

IPC : As for pipeline performance, typically IPC is used. It stands for "Instructions per cycle". A higher IPC value usually means a better throughput.

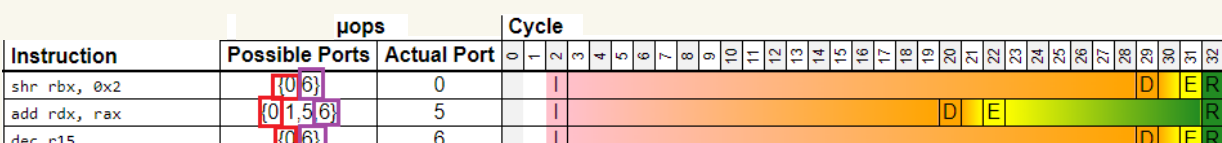
You can measure IPC with perf: <https://perf.wiki.kernel.org/index.php/Tutorial>

Rate of retired instructions : Apart from IPC, number of retired instructions should be checked. Retired instructions are not committed/fused 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.



Instruction lifecycle states in UCCA diagrams

CONTENTION FOR EXECUTION PORTS IN THE PIPELINE

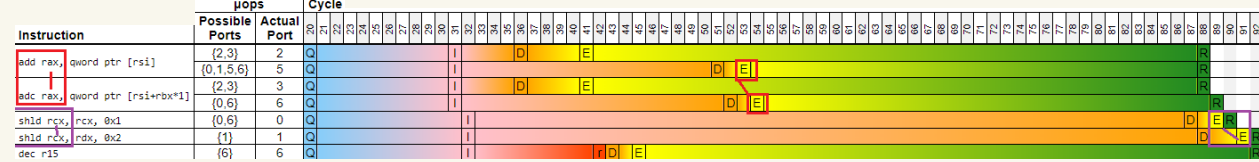


In the example above, all instructions are working on different registers, but SHR, ADD, DEC instructions are competing for ports 0 and 6. SHR and DEC are getting executed after ADD instruction.

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 Bashkatov's article](https://en.wikichip.org/wiki/skylake_server)

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.

Reference : [Denis Bashkatov's article](https://en.wikichip.org/wiki/skylake_server)

RDTSOP INSTRUCTION FOR MEASUREMENTS

RDTSOP instruction can flush the pipeline to discard the instructions prior to the measurement and read the TSC value of the CPU.

TSC : timestamp counter

You can use CPUID and RDTSOP combination in older systems that don't support RDTSOP.

ESTIMATING INSTRUCTION LATENCIES

Based on Agner Fog's [Instruction tables](https://www.agner-fog.com/instruction-tables), RDTSOP reciprocal throughput (clock cycle per instruction) is 32 on Skylake microarchitecture:

-> 1 cycle @4.5GHz = 0.22 nanoseconds
-> 32*0.22=7.04 nanoseconds

So its resolution estimate is about 7 nanoseconds on a 4.5 GHz Skylake microarchitecture. You have to recalculate it for different microarchitectures and clock speeds.

HYPERTHREADING / SIMULTANEOUS MULTITHREADING

Based on [Intel Software Developer's Manual Volume3](https://www.agner-fog.com/instruction-tables), it is implemented by 2 virtual cores that share resources including cache memory, branch prediction resources and execution ports. And AMD seems to use the resources in the same way based on Agner Fog's [microarchitecture book](https://www.agner-fog.com/instruction-tables).

For ex if you app is data-intensive, halved caches won't help. 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.

Note: Its generic name is simultaneous multithreading. Hyperthreading name used by Intel only.

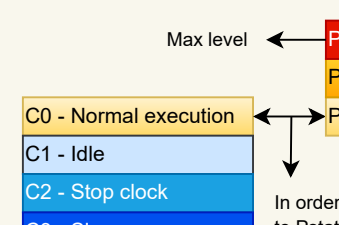
DYNAMIC CLOCK SPEEDS

Modern CPUs employ dynamic frequency scaling which means there is a min and max frequency per CPU core.

Also [ACPI](https://www.agner-fog.com/instruction-tables) defines multiple power states and modern CPUs implement those. P-States are for performance and C-States are for energy efficiency.

You can use Intel's [TurboBoost](https://www.intel.com/content/www/us/en/processors/core-i7-12th-gen/intel-processor-activity-monitor.html) or AMD's [TurboCore](https://www.amd.com/en/techcenter/boost) to maximise the CPU usage.

Note that SSE usage may also introduce downclocking, therefore they should be used carefully : [Daniel Lemire's article](https://en.wikichip.org/wiki/skylake_server)



ARITHMETIC REALM

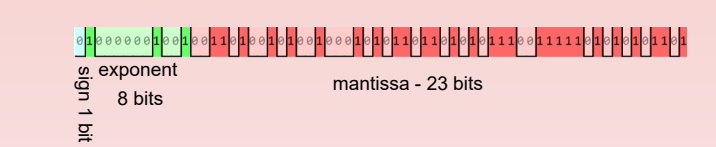
ARITHMETIC INSTRUCTION LATENCIES

You can see a set of arithmetic operations from fast to slow below.

The clock cycles are based on Agner Fog's Instruction tables & Skylake architecture on 64 bit registers.	
Bitwise operations	Integer add/sub : 0.25 to 1 clock cycle
	Floating point add : 3 clock cycles
	Floating point multiplication : about 5 clock cycles
	Floating point division : about 14-16 clock cycles
	Integer division : 24-90 clock cycles

FLOATING POINTS

X86 uses [IEEE 754](https://en.wikichip.org/wiki/skylake_server) standard for floating points. A 32 bit floating point consists of 3 parts in the memory layout. Below you can see all bits of 32bit IEEE 754 FP number. Used https://en.wikichip.org/wiki/skylake_server as visualizer :



A floating point's value is calculated as : $mantissa \times 2^{exponent}$

IEEE754 also defines [denormal numbers](https://en.wikichip.org/wiki/skylake_server). They are very small / near zero numbers.

The clock cycles are based on Agner Fog's [Instruction tables](https://www.agner-fog.com/instruction-tables) & Skylake architecture on 64 bit registers.

As floating points are approximations, denormal numbers are needed to avoid an undefined case of x+0, but x<0.

Without denormal the code to the right would make a divide-by-zero exception. Reference : [Blaze Dawson's article](https://en.wikichip.org/wiki/skylake_server)

Based on Agner Fog's [microarchitecture book](https://www.agner-fog.com/instruction-tables), Intel CPUs have a penalty for denormal numbers, for ex 120 clock cycles on Skylake. They also can be turned on of Intel CPUs.

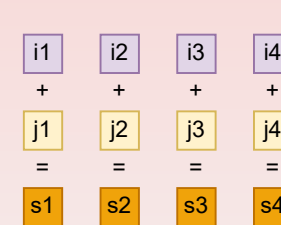
As for AMD side, the recent Zen architecture CPUs seemingly don't have the same performance degradation.

X86 EXTENSIONS

X86 extensions are specialised instructions. They have various categories such as [conditional and result-related operations](https://en.wikichip.org/wiki/skylake_server).

For the list of extensions : https://en.wikichip.org/wiki/skylake_server

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 :



In the example above, an array of 4 integers (11 to 44) are added to another array of integers (1 to 44). The result is also an array of sums (11 to 44). In this example, 4 additions are executed by a single instruction.

Some typical application areas are 3D graphics and quantitative finance.

Apart from arithmetic operations, they can be utilized for string operations as well : https://en.wikichip.org/wiki/skylake_server

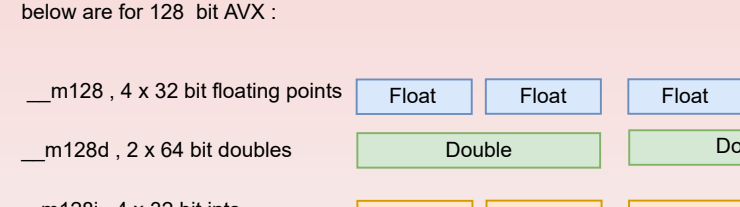
A SIMD based JSOM parser https://en.wikichip.org/wiki/skylake_server

X86 EXTENSIONS : SIMD DETAILS

The most recent SIMD instruction sets and their corresponding registers are :

AVX : 128 bits, XMM registers
AVX2 : 256 bits, YMM registers
AVX512 : 512 bits, ZMM registers

As for programming, there are also wider data types. The data type diagrams below are for 128 bit AVX :



Note that SSE instructions require more power, therefore their usage may also introduce downclocking. They should be benchmarked : [Daniel Lemire's article](https://en.wikichip.org/wiki/skylake_server)

LOAD STORE REALM

LOAD & STORE BUFFERS

Load and store buffers allow CPU to do out-of-order execution on loads and stores by decoupling speculative execution and committing the results to the cache memory.

Reference : https://en.wikimedia.org/wiki/Memory_Disambiguation

STORE-TO-LOAD FORWARDING

Using buffers for stores and loads to support out of order execution leads to a data synchronization issue. That issue is described in en.wikimedia.org/wiki/Memory_DisambiguationStore_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 :

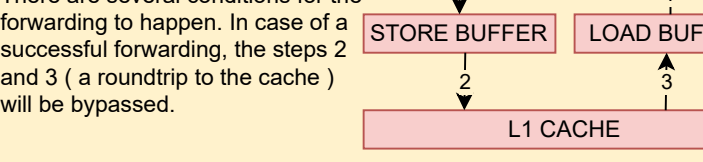
```
mov [eax], ecx ; STORE : Write the value of ECX register to the memory address which is stored in EAX register
mov ecx, [eax] ; LOAD : Read the value from that memory address (which was just used) and write it to ECX register
```

STORE-TO-LOAD FORWARDING & LHS & PERFORMANCE

Based on [Intel Optimization Manual 3.6.4](https://en.wikichip.org/wiki/skylake_server), 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.wikimedia.org/wiki/Load_Hit_Store

There are several conditions for the forwarding to happen. In case of a successful forwarding, the steps 2 and 3 (a roundtrip to the cache) will be bypassed.



The conditions for a successful forwarding and latency penalties in case of no-forwarding can be found in Agner Fog's [microarchitecture book](https://www.agner-fog.com/instruction-tables).

Previous game consoles PlayStation3 and Xbox360 had PowerPC based processors which did in-order execution rather than out-of-order execution. Therefore developers had to separately handle LHS by using [flushing](https://en.wikichip.org/wiki/skylake_server) keyword and other methods : [Alan Rusakov's article](https://en.wikichip.org/wiki/skylake_server)

BRANCH PREDICTION REALM

BRANCH PREDICTION BASICS

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

Goal if predicted correctly : If the right branch was predicted that will increase the throughput as it completed fetching a set of instructions in advance.

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

What are branch instructions instructions ? : Unconditional ones (jmp), conditional ones (eg: jne) : called

How : There are auxiliary hardware buffers.

Branch target buffer stores target addresses / instruction pointers of branches. AMD uses multiple level of BTBs : L1 BTB, L2 BTB etc.

Pattern history tables track the history of results (whether it was taken or not) per branch.

A hypothetical pattern history table

BTB methods : 2-LEVEL ADAPTIVE BRANCH PREDICTION

Saturating counter
A 2-bit saturating counter can store 4 strength states.

Whenever a branch is taken it goes stronger.

And whenever a branch is not taken it goes weaker.

2 level adaptive predictor

In this method, you store the history of last n occurrences in a history register which is n bits.

Also you create a table called "system history table" for that branch. That pattern history table keeps 2^n rows and each row has a saturating counter.

The branch history register will be used to choose which row will be used from the pattern history table.

Reference : Agner Fog's [microarchitecture book](https://www.agner-fog.com/instruction-tables)

BP METHODS : AMD PERCEPTIONS

They are used in Zen architectures.

A [perception](https://en.wikichip.org/wiki/skylake_server) is basically the simplest form of machine learning. They can be considered as a linear array of weights.

Agner Fog mentions that they are good at predicting very long branches compared to 2-level adaptive branch prediction in his [microarchitecture book](https://en.wikichip.org/wiki/skylake_server).

For details of perception based branch prediction : [Dynamic Branch Prediction with Perceptions by Daniel Amores and Calvin Lin](https://en.wikichip.org/wiki/skylake_server)

The output "Y" (in this case whether a branch taken or not) is calculated by dot product of the weight vector and the input vector.

Intel LSD (LOOP STREAM DETECTOR)

Intel LSD will detect a loop and stop fetching instruction to improve frontend bandwidth. Several conditions mentioned in [Intel Optimization Manual](https://en.wikichip.org/wiki/skylake_server).

- Loop body size up to 60 uops, 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 10 iterations.

Note that LSD is disabled on Skylake Server CPUs. Reference : https://en.wikichip.org/wiki/skylake_server

You can consider disabling system patches for speculative execution related vulnerabilities such as Meltdown and Spectre for performance, if it is disable in your system.

Kernel.org documentation : https://en.wikichip.org/wiki/skylake_server

Meltdown paper : https://en.wikichip.org/wiki/skylake_server

Spectre paper : https://en.wikichip.org/wiki/skylake_server

ESTIMATED LIMITS : HOW MANY IFs ARE TOO MANY ?

As for max number of writes in BTBs, there are estimations made by stress testing the BTB with conditions of branch instructions.

Intel Xeon Gold 6262 -> roughly 4K
AMD EPYC 7713 -> roughly 3K

Reference : [Mark Malozki's article on Cloudflare blog](https://en.wikichip.org/wiki/skylake_server)

VIRTUAL MEMORY REALM

VIRTUAL MEMORY ORGANISATION

Why virtual memory ?
Because cumulative memory requirement of multiple processes in an OS can be more than the system capacity.

It is basically for sharing memory resources between multiple processes.

It also provides security by isolating processes.

Pages
- Minimum addressable virtual space that can be requested from OS.

- Typically 4KB

Swapping
Happens when the page is not on physical memory but on the swap file which is on the harddrive.

Page faults
Happens when the page is not on physical memory but on the swap file which is on the harddrive.

VIRTUAL ADDRESS SPACE

SYSTEM MEMORY ADDRESS SPACE

TLB pressure

If page pressure is 4K, that increases the CPU support for larger pages.

x86-64 CPUs support huge pages from 2MB to 1GB to reduce the pressure on TLB.

OS support

Linux implementation refers to them as huge pages and Windows calls them as large pages.

You shall check your OS and CPU in combination to find out the supported sizes.

Regular pages

Huge pages

TLB PRESSURE & HUGE PAGES

TLB pressure

If page pressure is 4K, that increases the CPU support for larger pages.

x86-64 CPUs support huge pages from 2MB to 1GB to reduce the pressure on TLB.

OS support

Linux implementation refers to them as huge pages and Windows calls them as large pages.

You shall check your OS and CPU in combination to find out the supported sizes.

Regular pages

Huge pages

VIRTUAL MEMORY ADDRESS TRANSLATION

Address translation & Page table

CPU with virtual addresses and those addresses need to be converted to physical addresses.

Page table structures on system memory are used for this purpose.

TLB (Translation lookaside buffer)

TLBs are caches in CPUs to make the translation process faster. Modern CPUs have multiple levels of TLBs.

(Intel refers to L2 TLB as sTLB)

TLB Shootdowns

See the shutdown realm below.

ITLB

Apart from data TLB, there is also ITLB for caching addresses of instructions on both Intel and AMD architectures.

PAGE TABLE WALKING

Even with pages which group addresses, having all pages in a page table would still need too much storage on 64-bit systems. Therefore page tables are implemented hierarchically.

Memory is divided into address spaces. And there is a tree data structure for each address space in the page table. Processes have to walk the page table level by level in the hierarchy to find out the final actual address.

4 level page tables is the most common one. In the diagram above first 48 bits of a 64 bit address are used for page table walking. All of 48 bits has to be used in order to find out the final actual address.

Intel CPUs started to support 5 level tables since Ice Lake.

The advantage of another level is that you can address even more space.

The disadvantage is that the time needed to walk the page table increases due to a new level of indirection.

Page table structures on system memory are used for this purpose.

TLB (Translation lookaside buffer)

TLBs are caches in CPUs to make the translation process faster. Modern CPUs have multiple levels of TLBs.

(Intel refers to L2 TLB as sTLB)

TLB Shootdowns

See the shutdown realm below.

ITLB

Apart from data TLB, there is also ITLB for caching addresses of instructions on both Intel and AMD architectures.

SYSTEM MEMORY REALM

DDR RAMs

DDR RAMs are the most common commodity hardware as system memory.

They are found in forms of DIMMs (Dual In-line Memory Modules) / RAMS.

Organisation

System memory / RAM is organised as collection of ranks.

Each rank have banks which are collection of DRAM cells per bit.

DRAM refreshes

DRAM circuits use capacitors which lose their charge over time. (See the cache memory realm). So RAMs have to refresh their DRAM cells periodically.

As for DDR4, refreshing is rank-level which means the other banks in the same rank become inaccessible. DDR5 comes with [adaptive bank-refresh](https://en.wikichip.org/wiki/skylake_server) feature which allows a more fine-grained bank-level refresh. Therefore it can offer a higher throughput.

DRAM refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity

DDR5 refresh granularity