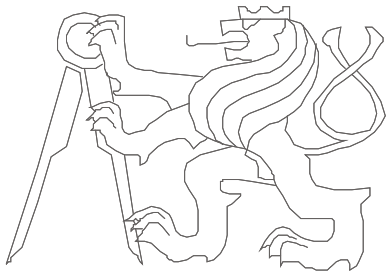
Pokročilé architektury počítačů

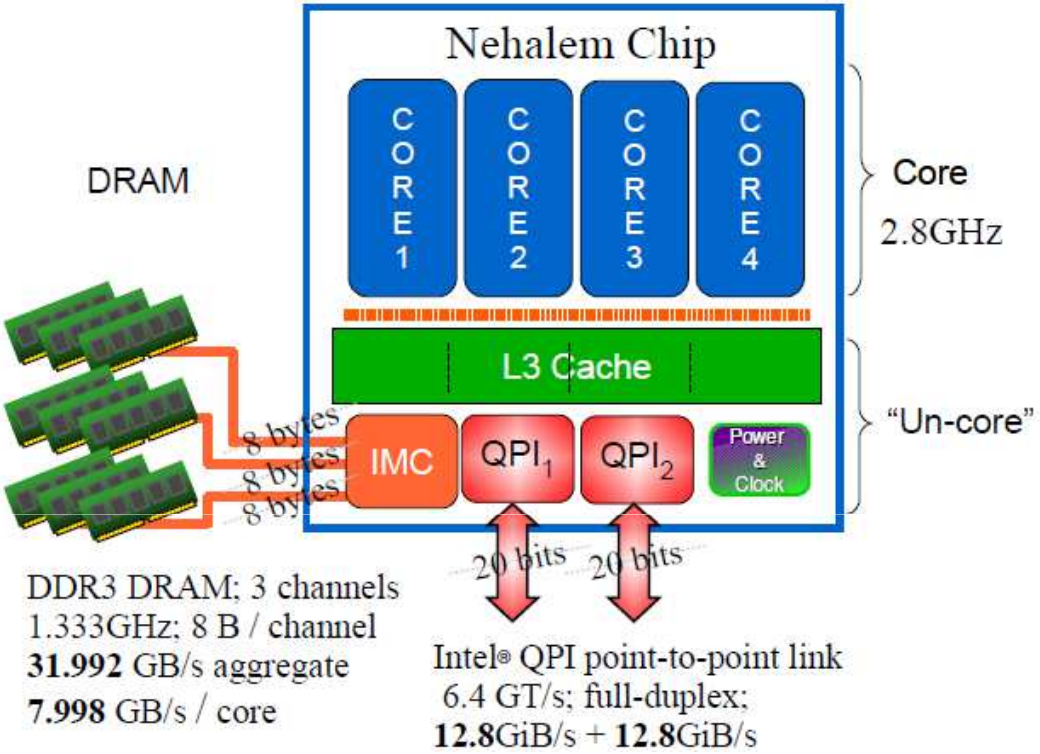
Teorie v praxi…

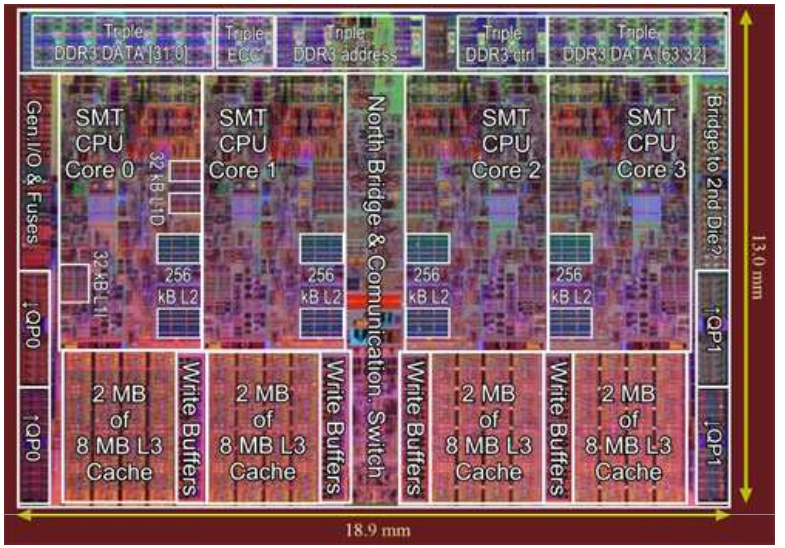
Intel Nehalem (2008) – první generace Core i5 a Core i7



České vysoké učení technické, Fakulta elektrotechnická

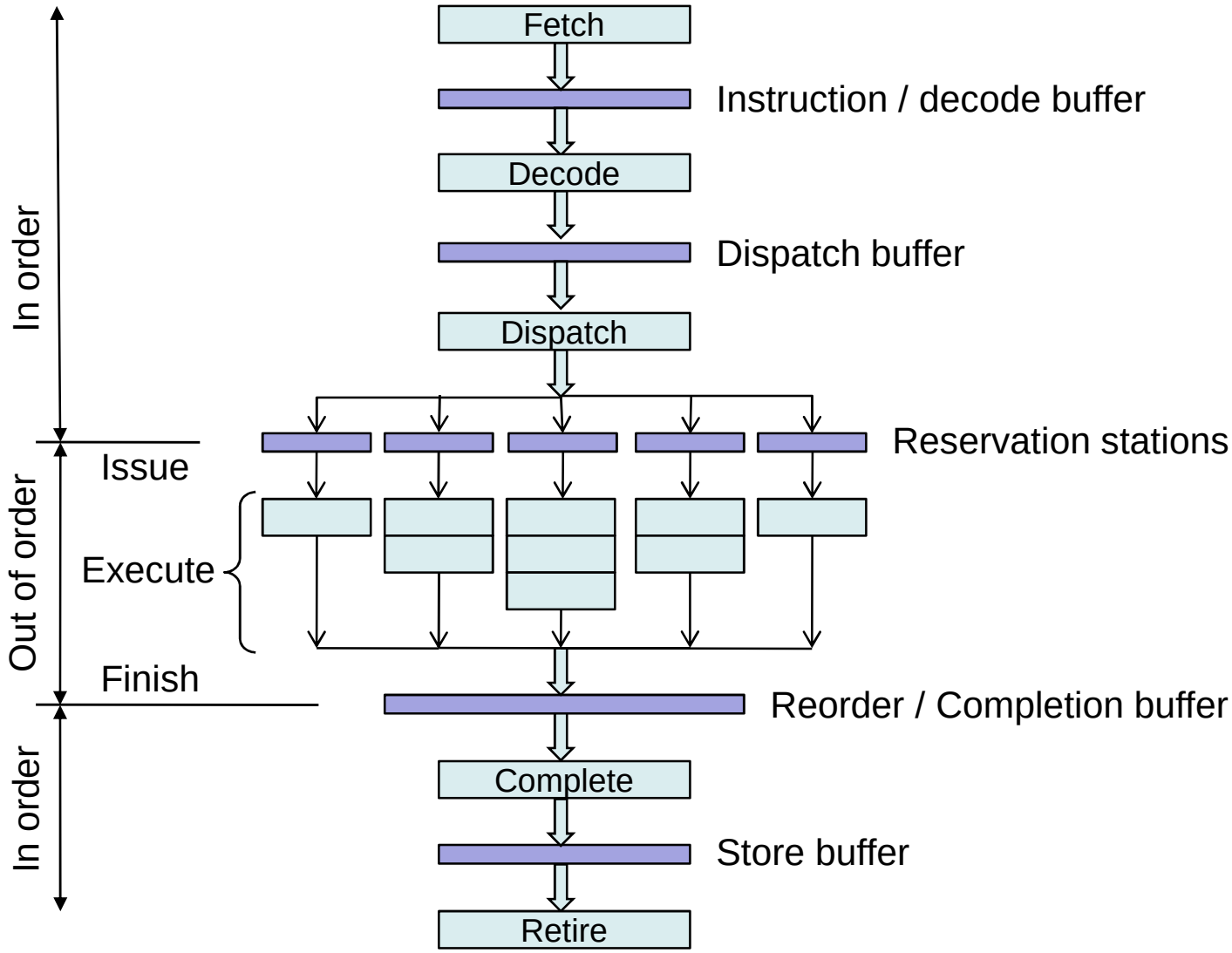
Intel Nehalem



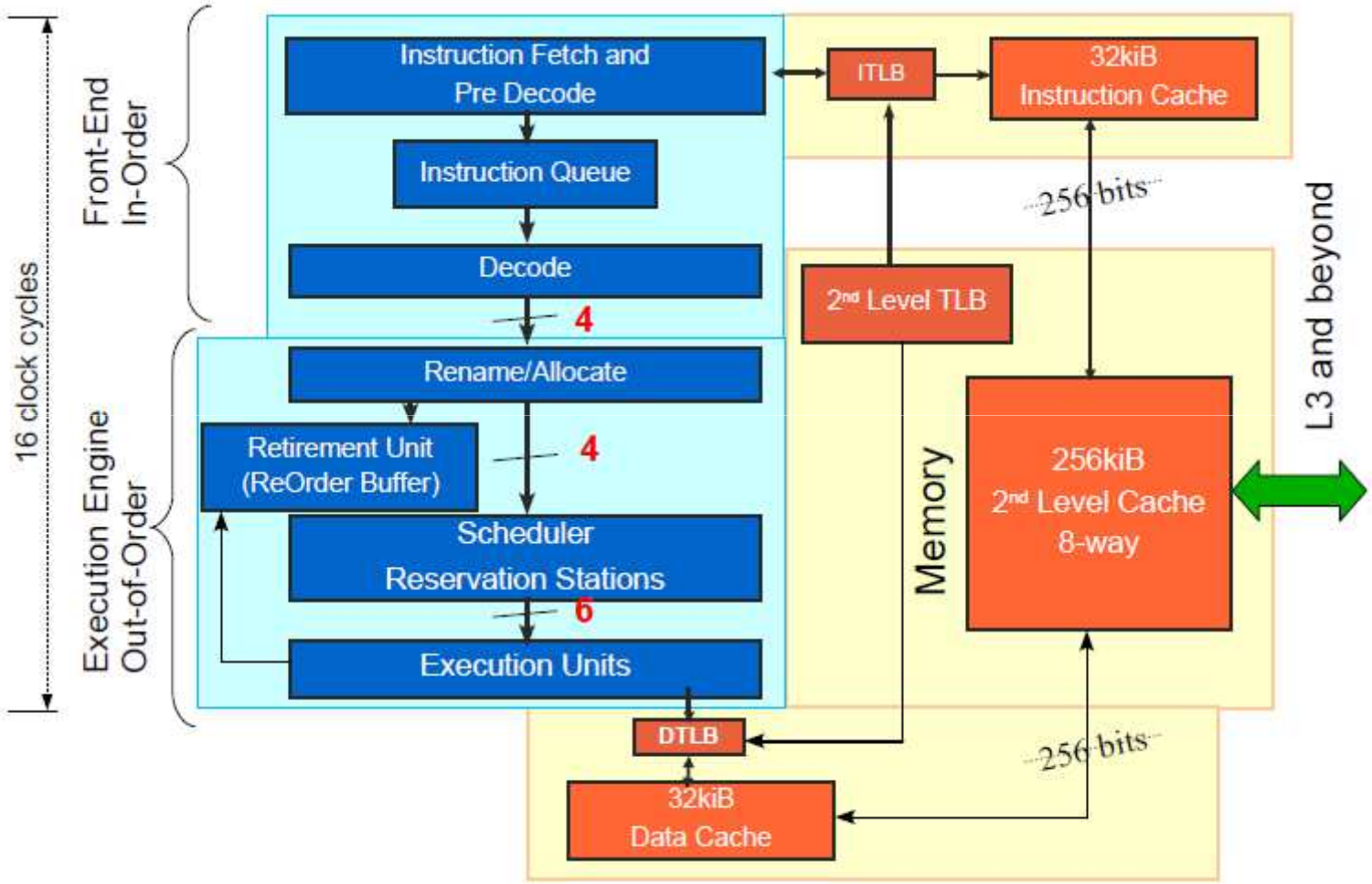


- core / un-core
- UIU: Un-Core Interface Unit (switch connecting the 4 cores to the 4 L3 cache segments, the IMC and QPI ports),
- IMC: 1 integrated memory controller with 3 DDR3 memory channels,
- QPI: 2 Quick-Path Interconnect ports
- dodatečné obvody pro zajištění koherence cache, řízení spotřeby, monitorování výkonu

Pamatujete si?

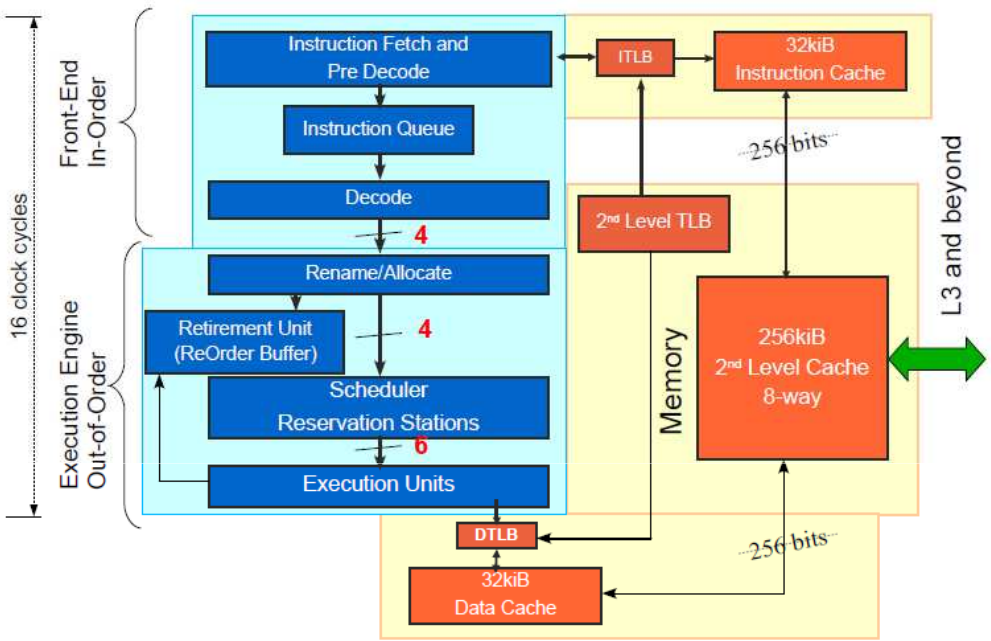


Nehalem Core Pipeline



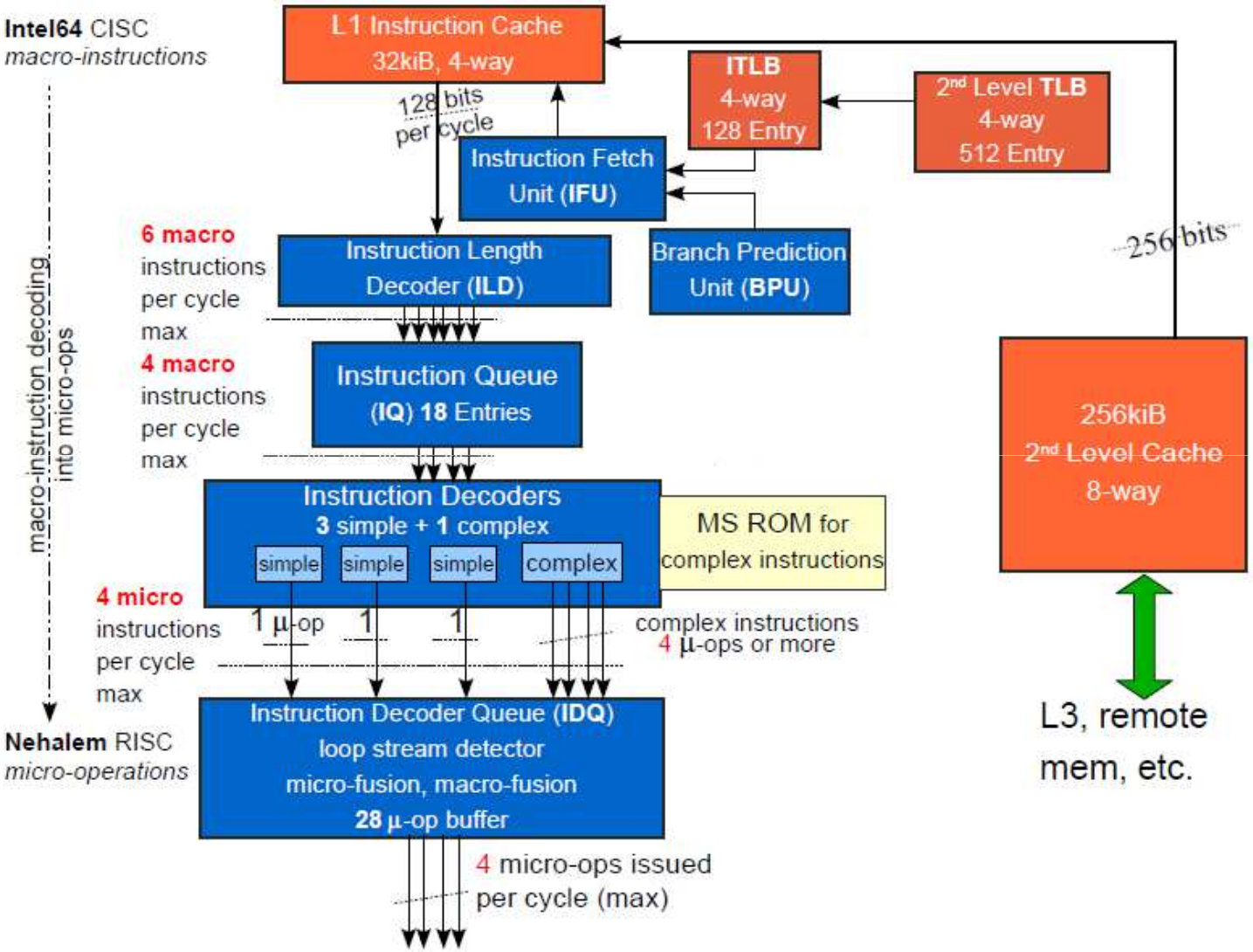
Nehalem Core Pipeline

- 14-stage core pipeline
- Front-End Pipeline (FEP)
  - překlad makroinstrukcí na mikroinstrukce
- Execution Engine (EE)
  - dynamické rozvrhování a dispečování mikroinstrukcí
- Retirement Unit (RU)
  - stav procesoru je aktualizován v původním pořadí



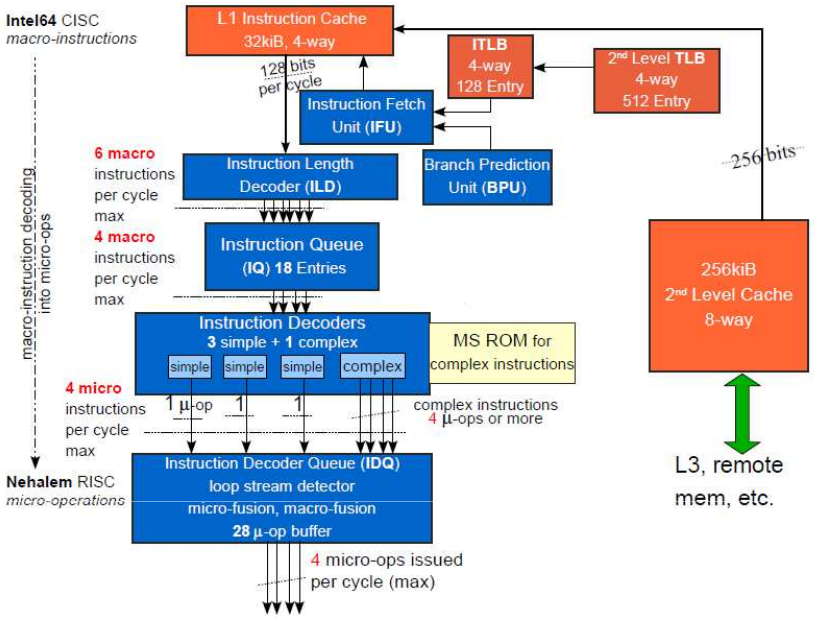
- TLB (Translation Lookaside Buffer) - podpora překladu virtuálních adres na fyzické; obsahuje záznamy ze stránkovacích tabulek. A co přepínání kontextu?? ASN (Address Space Number), (PGE flag)

Front-End Pipeline (FEP)



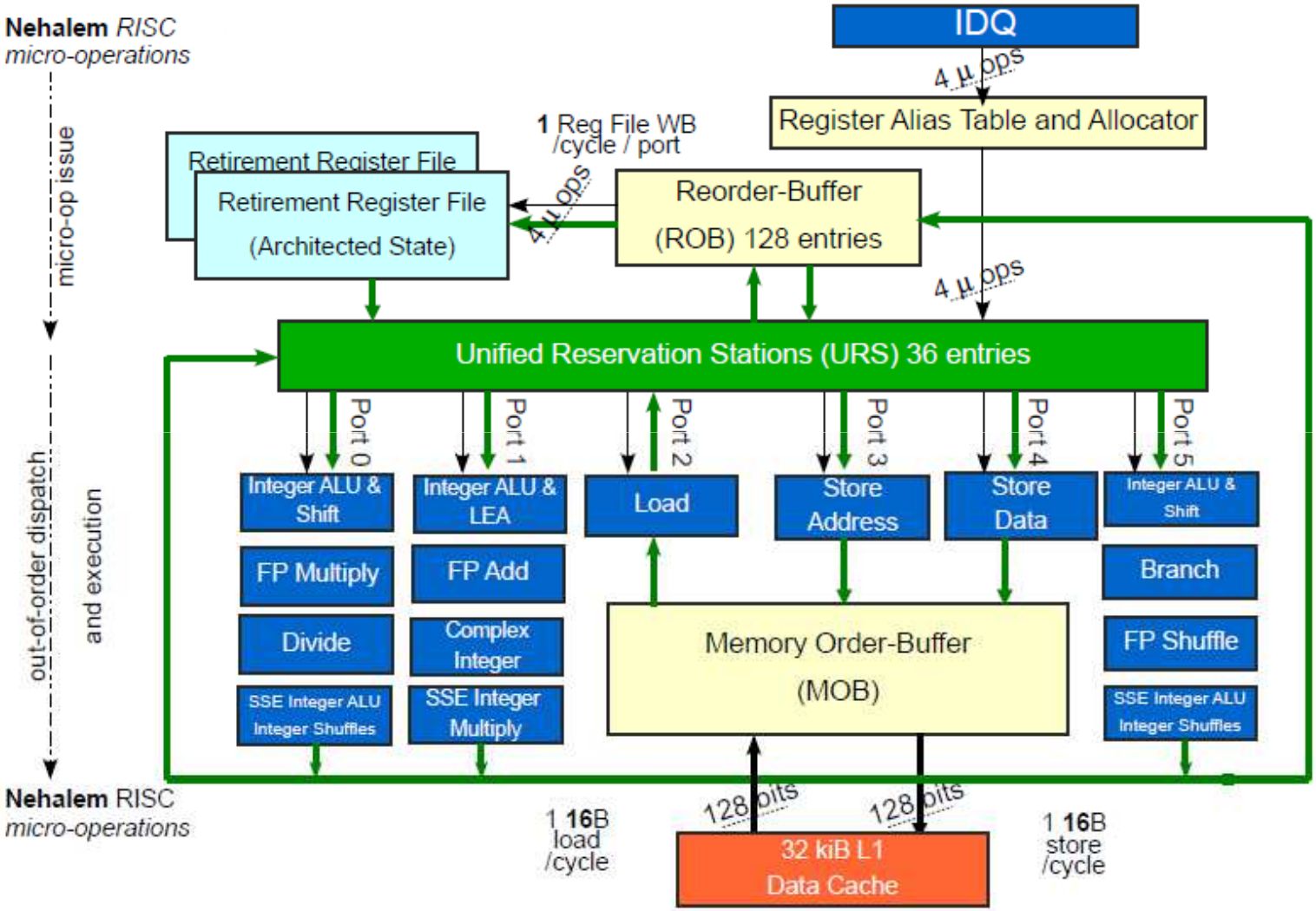
Front-End Pipeline (FEP) – několik poznámek

- všechny IDU (Instruction Decoding Unit) podporují časté případy instrukčního toku (mikro-fusion, macro-fusion, stack pointer tracking)
- LSD (**Loop Stream Detection**) redukuje také ztráty z důvodu nezarovnaných instrukcí, LCP (Length Changing Prefixes) pokuty, spotřebu energie



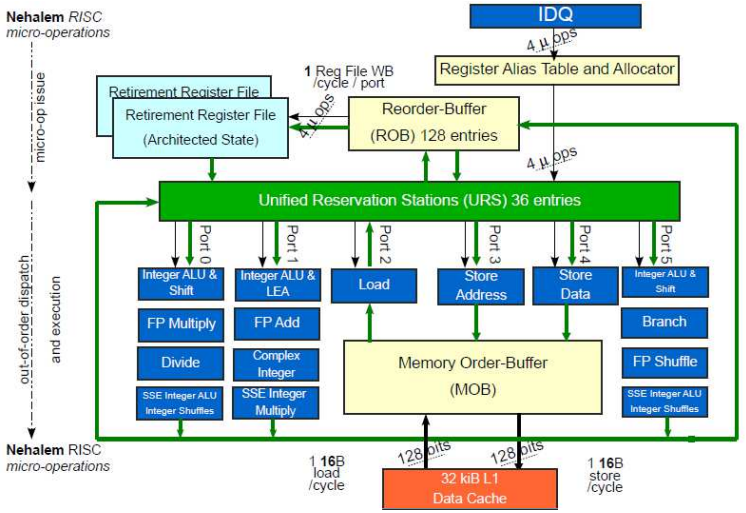
- SPT (**Stack Pointer Tracking**) implementuje Stack Pointer Register update logiku pro instrukce manipulující se Stack-em (push, pop, call, leave, ret) – tyto makroinstrukce dřív zabírali několik mikro-ops, nyní jedna mikro-op.
- **Mikro-fusion** seskupuje několik mikroinstrukcí téže instrukce do jedné komplexní mikroinstrukce – redukuje celkové „bit-toggling“
- **Macro-fusion** seskupuje sousedící makroinstrukce do jedné mikroinstrukce (logické porovnání, test + podmíněný skok)

Execution Engine (EE)



Execution Engine (EE) – několik poznámek

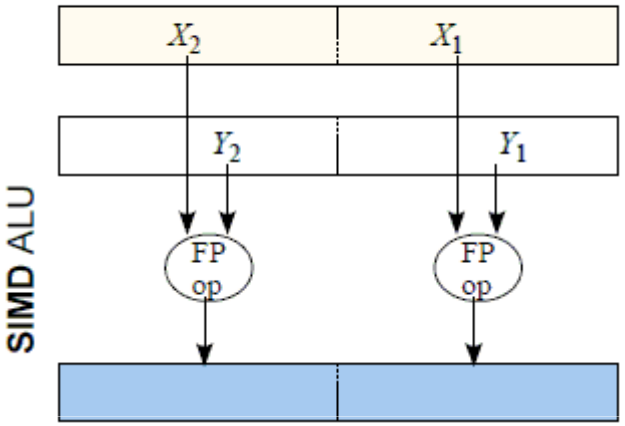
- MOB (Memory Order Buffer) podporuje spekulativní čtení (a mimo pořadí) a zápis a zabezpečuje, že zápis do paměti se vykoná v správném pořadí a se správnými daty
- Přejmenování registrů
- Bypass network – forwarding



- Execution Units – 6 operací (3 paměťové, 3 aritmetické/logické) za jeden cyklus:
- **Port 0** podporuje: Integer ALU and Shift Units, Integer SIMD ALU and SIMD shuffle, Single precision FP MUL, double precision FP MUL, FP MUL (x87), FP/SIMD/SSE2 Move and Logic and FP Shuffle, DIV/SQRT
- **Port 1** podporuje: Integer ALU, integer LEA and integer MUL, Integer SIMD MUL, integer SIMD shift, PSAD and string compare, and FP ADD
- **Port 2** Integer loads, **Port 3** Store address, **Port 4** Store data
- **Port 5** podporuje: Integer ALU and Shift Units, jump, Integer SIMD ALU and SIMD shufflee, FP/SIMD/SSE2 Move and Logic

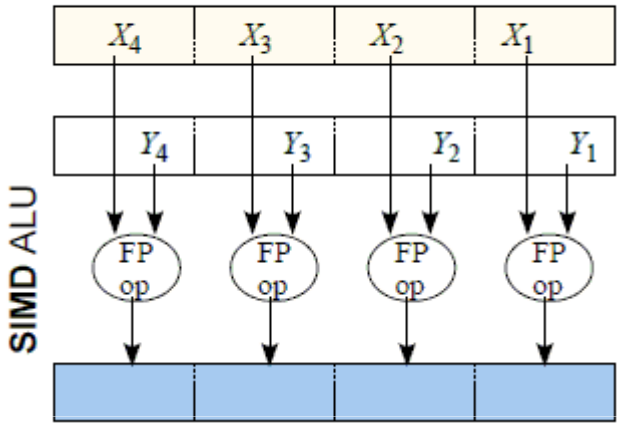
Execution Engine (EE) – několik poznámek

1 128 bit **XMM** data register ≡
2 packed DP FP quantities



2 64-bit **Double**-Precision
SIMD FP operations
with **XMM** data registers
↔
2 DP FP ops / cycle / port

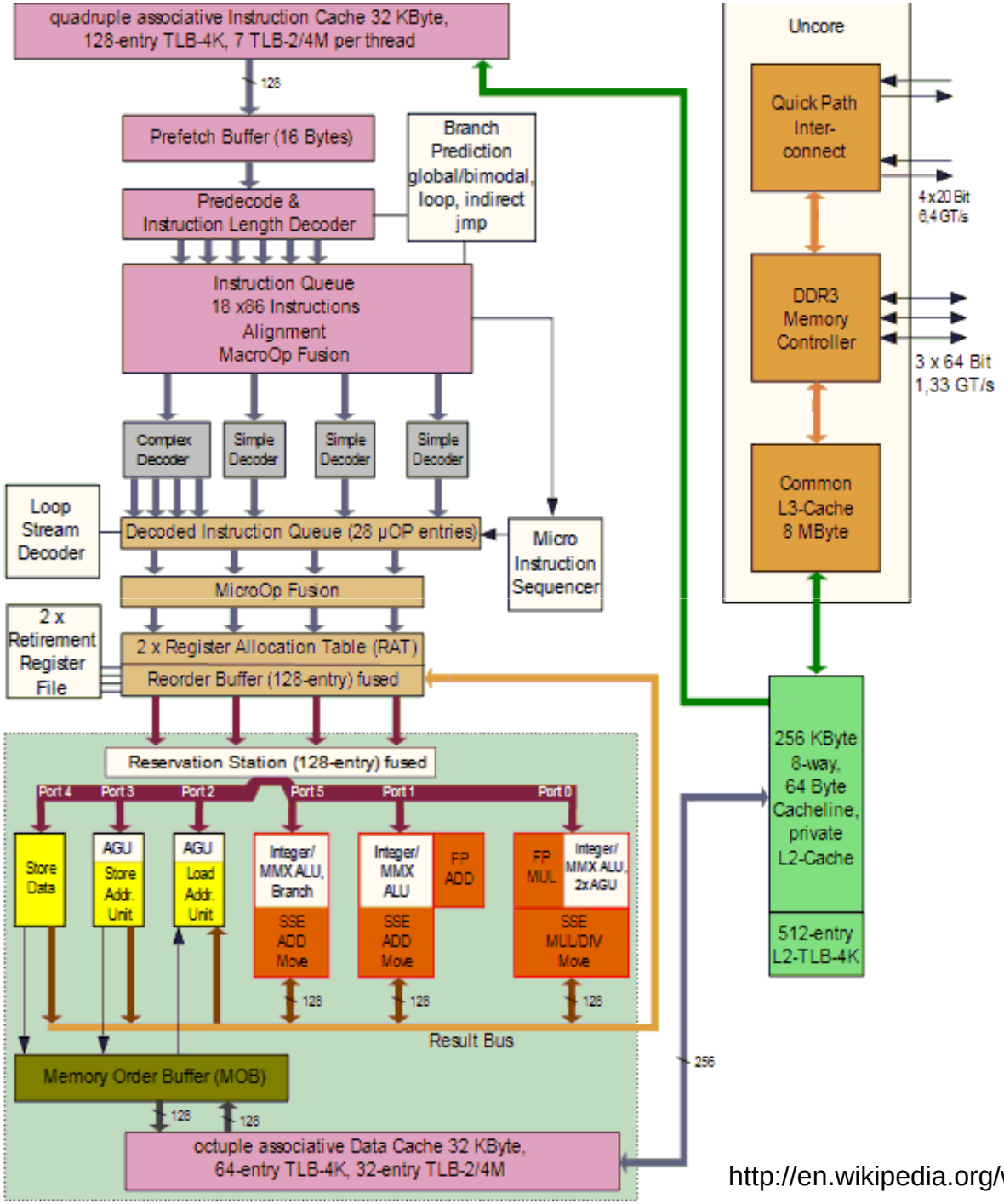
1 128 bit **XMM** data register ≡
4 packed SP FP quantities

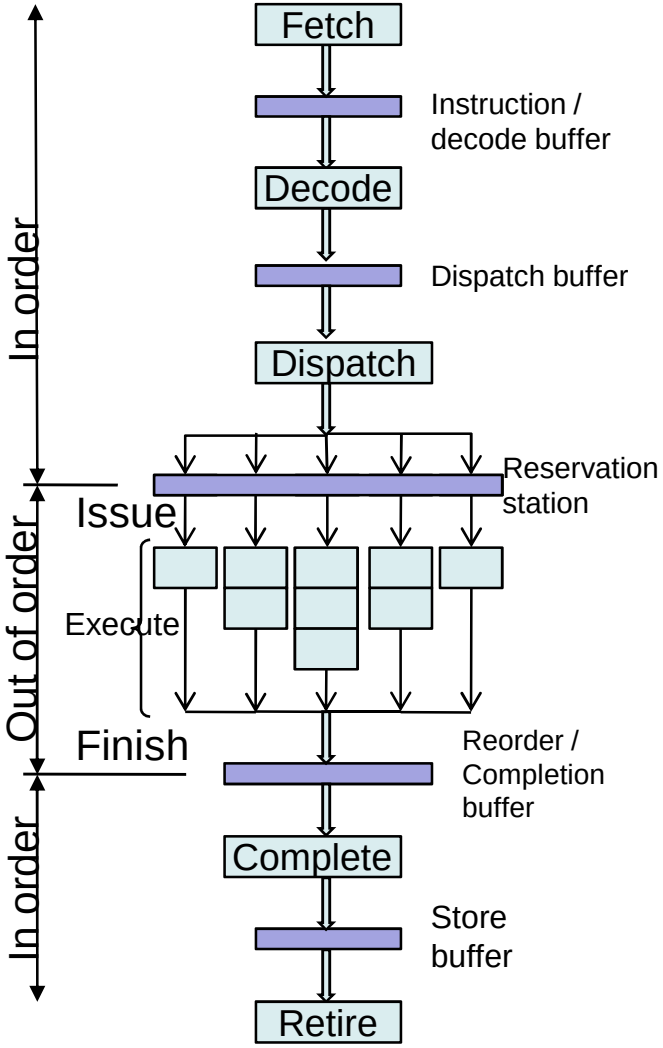


4 32-bit **Single**-Precision
SIMD FP operations
with **XMM** data registers
↔
4 SP FP ops / cycle / port

$$\begin{aligned}
\mathbf{11.2}\ \text{Giga FLOPs/sec/core} &= 2.8\text{GHz} \times 4\text{FLOPs/Hz} \\
\mathbf{44.8}\ \text{Giga FLOPs/sec/socket} &= 11.2\text{Giga FLOPs/sec/core} \times 4\text{cores} \\
\mathbf{89.6}\ \text{Giga FLOPs/sec/node} &= 44.8\text{Giga FLOPs/sec/socket} \times 2\text{sockets},
\end{aligned}$$

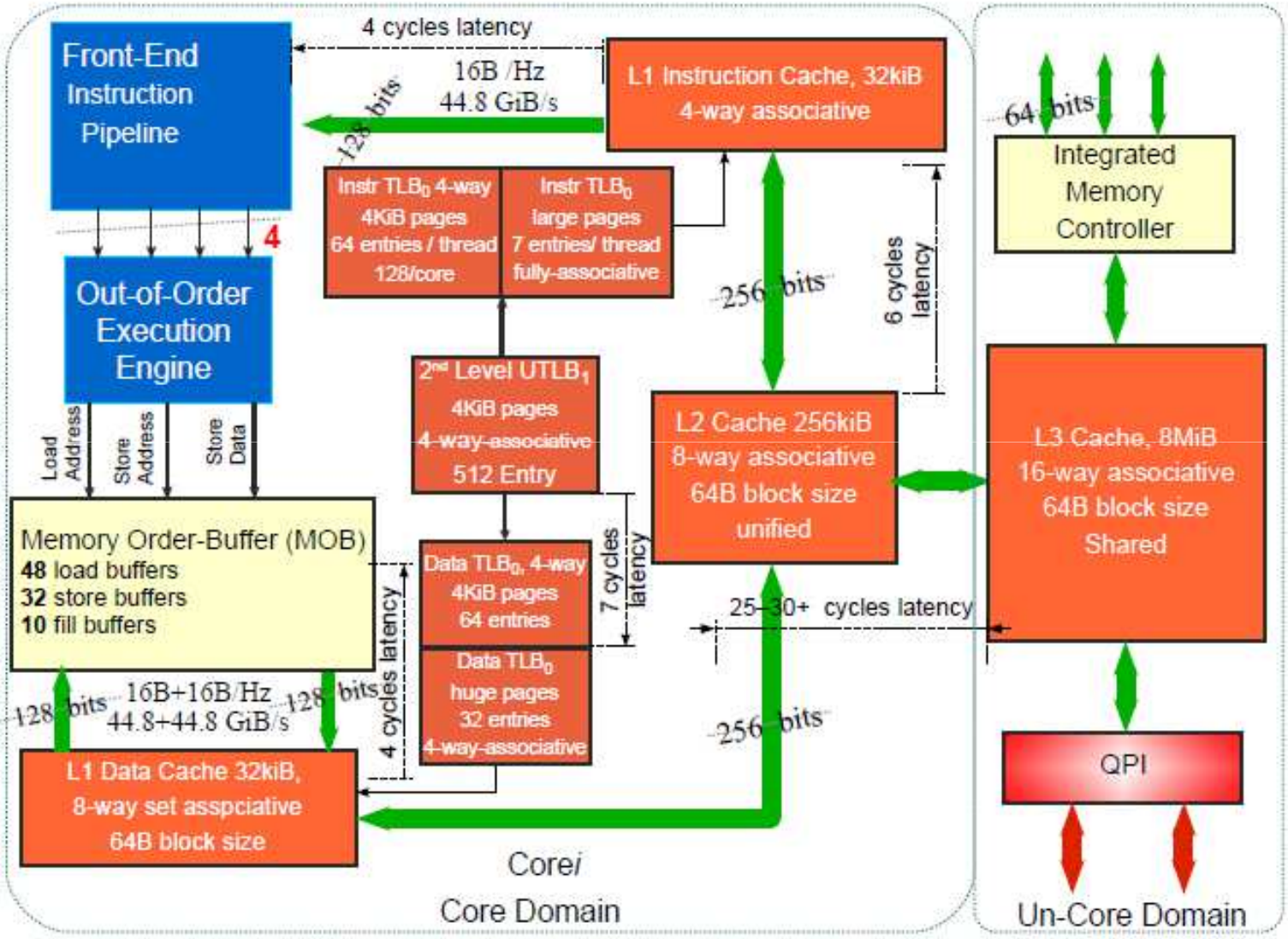
Vidíte podobnost?





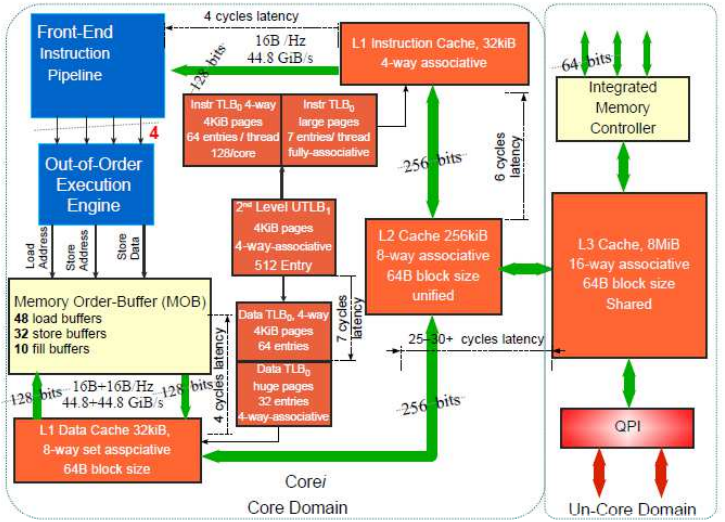
http://en.wikipedia.org/wiki/Nehalem_%28microarchitecture%29

Organizace paměti



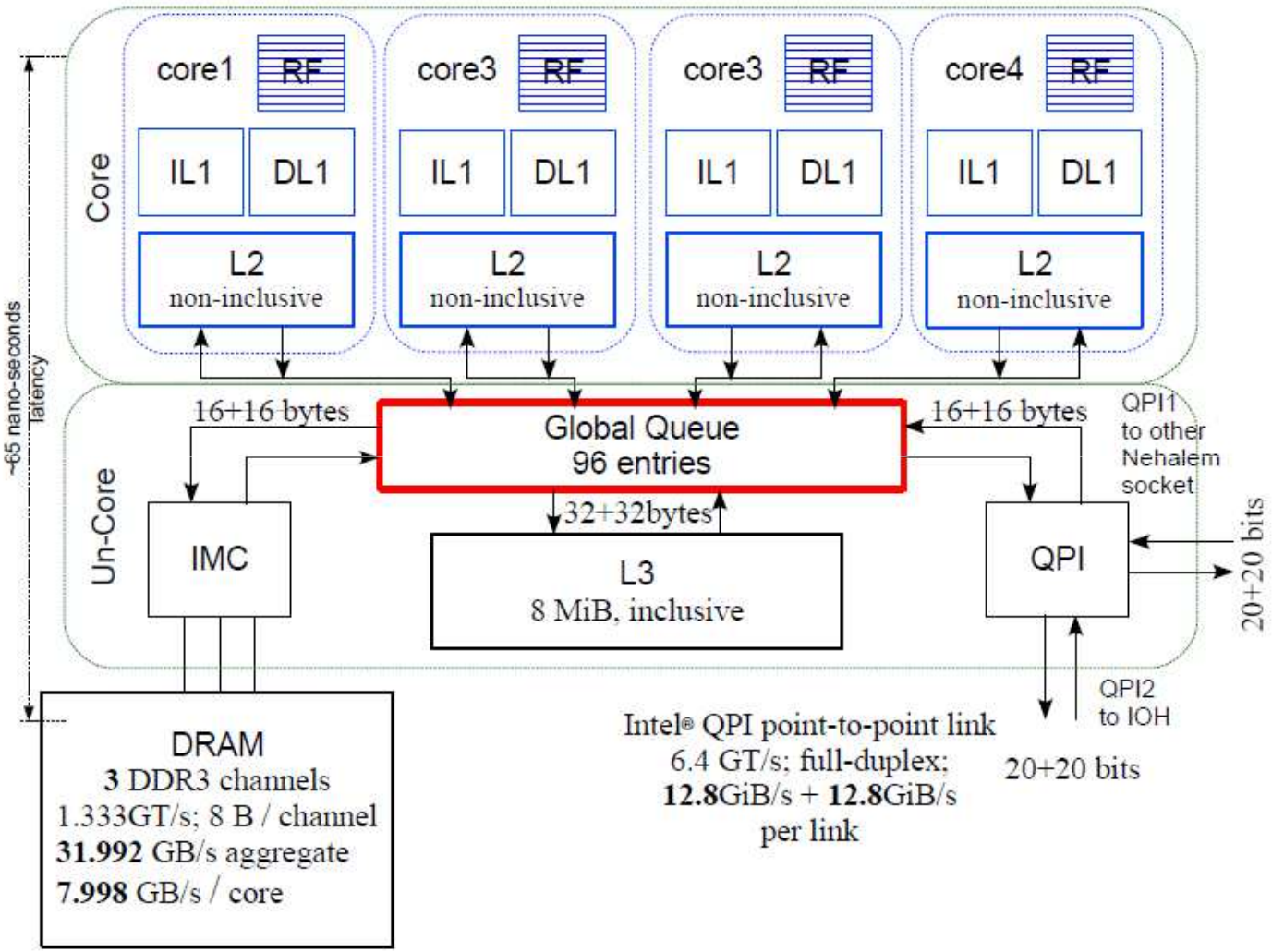
Organizace paměti – několik poznámek

- Velkost bloku: 64B
- procesor vždy čte řádek cache ze systémové paměti zarovnán na 64B a nepodporuje částečně plněné řádky
- L1 – Harvard. V SMT sdílená oběma vlákny, Instrukční – 4-way, Datová 8-way.
- L2 – unifikovaná, 8-way, neinkluzivní, WB
- L3 – unifikovaná, 16-way, inkluzivní
(řádek obsažen buď v L1 nebo L2 se nachází v L3), WB



- **Store Buffers** – dočasně uchovávají data pro každý zápis. Netřeba čekat na zápis do cache či paměti. Zajišťují, že zápisy jsou ve správném pořadí a také když je potřeba:
  - výjimka, přerušení, instrukce serializace, lock,..
  - store forwarding
- Data prefetching to L1 caches (streaming prefetcher, strided prefetcher)
- Data prefetching to L2 – na základě vzorů do L1 a L2

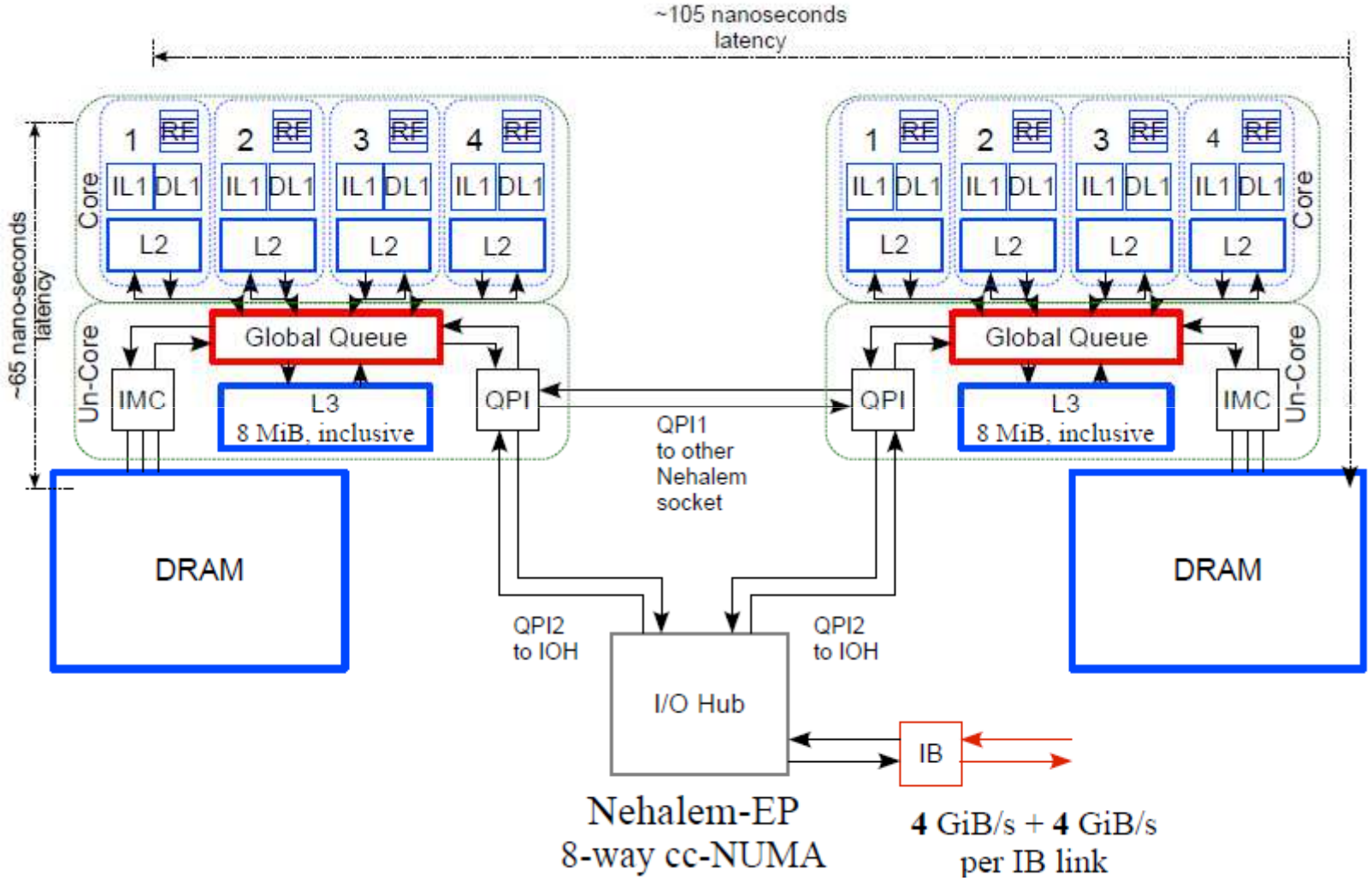
L3 – sdílená cache pro všechny jádra



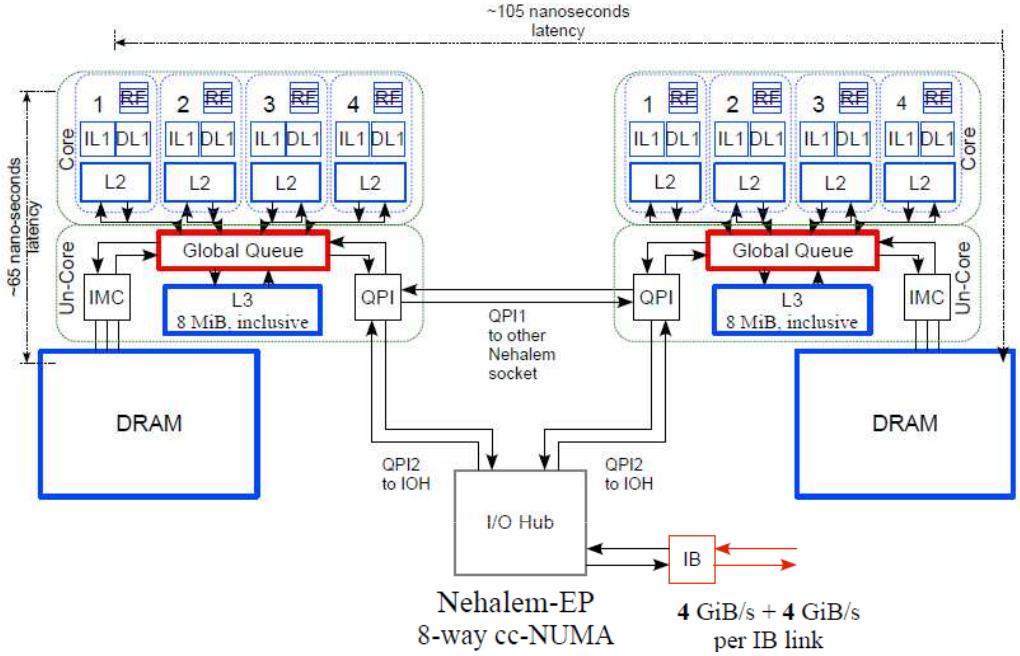
Global Queue

- GQ leží v části „uncore“ a řídí tok dat
- Žádosti o cache line z on-chip jádra, z remote procesoru, nebo z I/O hub-u
- MESIF protokol – využívá inkluzivní L3
- Cache line requests from the on-chip four cores, from a remote chip or the I/O hub are handled by the Global Queue (GQ) which resides in the Uncore. The GQ buffers, schedules and manages the flow of data traffic through the uncore.
- Write Queue (WQ): 16-entry queue pro zápis do paměti z lokálních jader
- Load Queue (LQ): 32-entry queue pro čtení z paměti z lokálních jader
- QPI Queue (QQ): 12-entry queue pro žádosti doručené přes QPI

Dva procesory



Dva procesory – přístup do lokální RAM

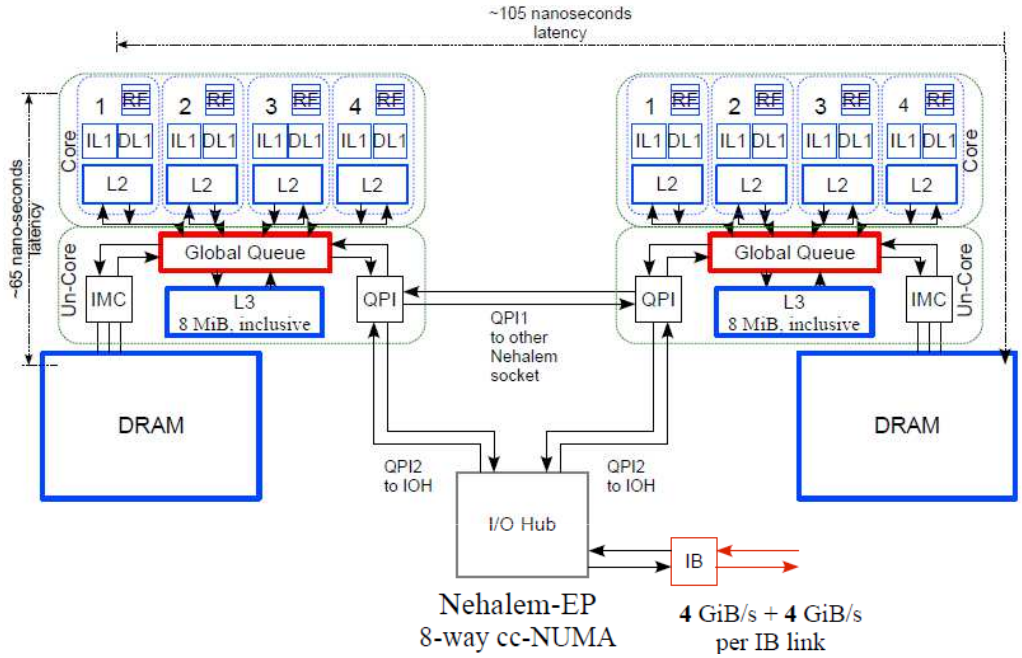


MESIF protokol

L3 cache – 4 bity

1. Procesor 0 žádá cache line, která se nenachází v L1, L2 ani v L3
   - Procesor 0 žádá data, která jsou v jeho DRAM,
   - Procesor 0 slídí jestli data neposkytl Procesor 1
2. Reakce
   - lokální DRAM vrátí data,
   - Procesor 1 vrátí odezvu,
   - Procesor 0 naplní cache

Dva procesory – přístup do vzdálené RAM



MESIF protokol

L3 cache – 4 bity

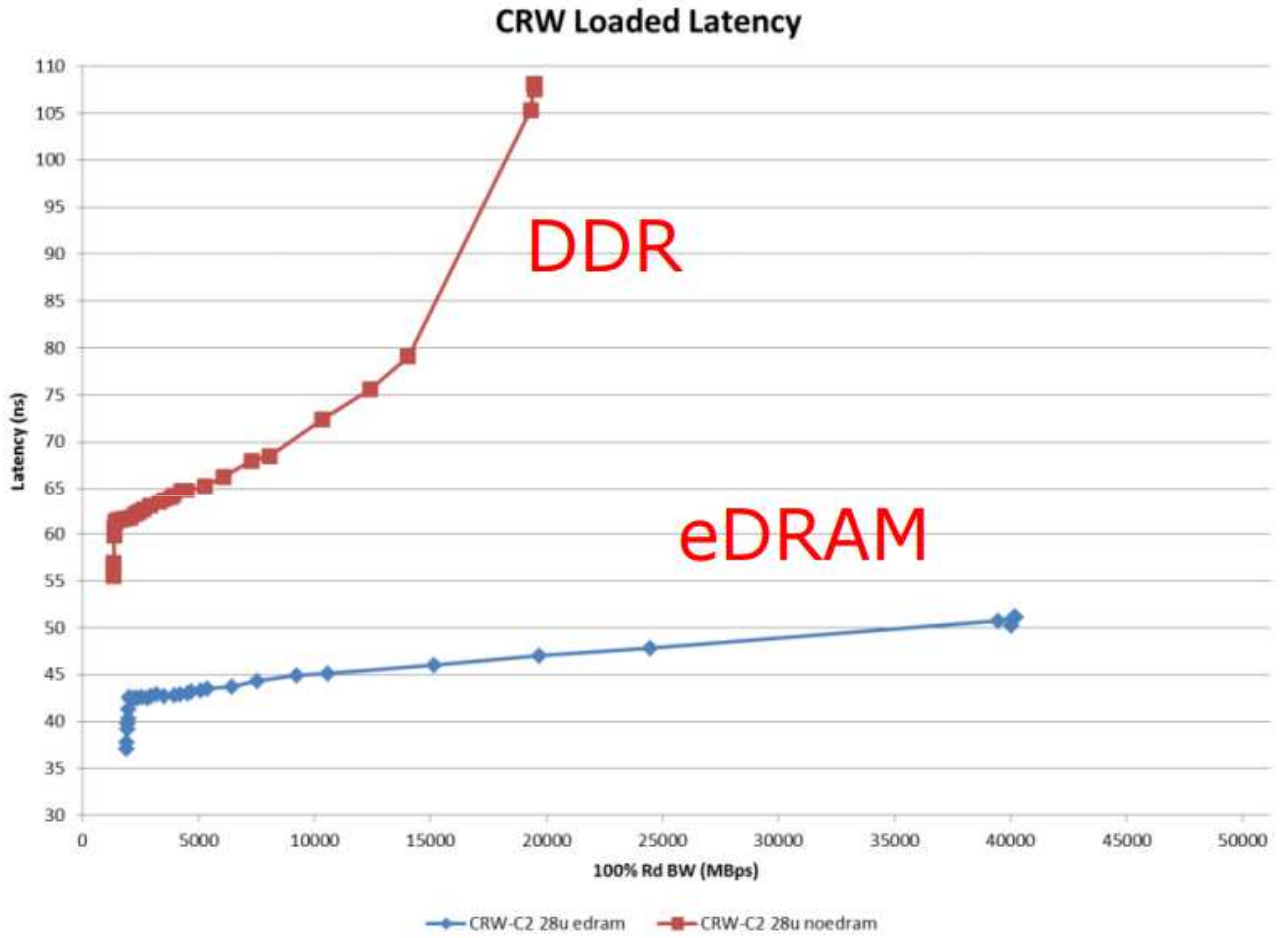
L3 Shared
L3 Modified

1. Procesor 0 žádá cache line, která není v jeho L1, L2 ani L3 cache
2. Požadavek se posílá přes QPI Procesoru 1
3. Procesor 1 reaguje (IMC generuje požadavek do DRAM, Procesor 1 zkouší jeho cache)
4. Odezva
   - Data se vrátí do Procesoru 0 přes QPI
   - Procesor 0 naplní cache a nastaví Shared nebo Exclusive

Haswell (2013)

- Čtvrtá generace Core i3, i5, a i7
- Zavádí další množství instrukcí,
  např. Fused Multiply-Subtract:
  A = AxB – C
  C -= AxB
- Současný trend: **System-on-a-Chip** (SoC) - CPU, GPU, Last Level Cache a systemové I/O na jednom čipu
- 128 MB L4 cache – eDRAM (Iris Pro model – integrovaná grafika)

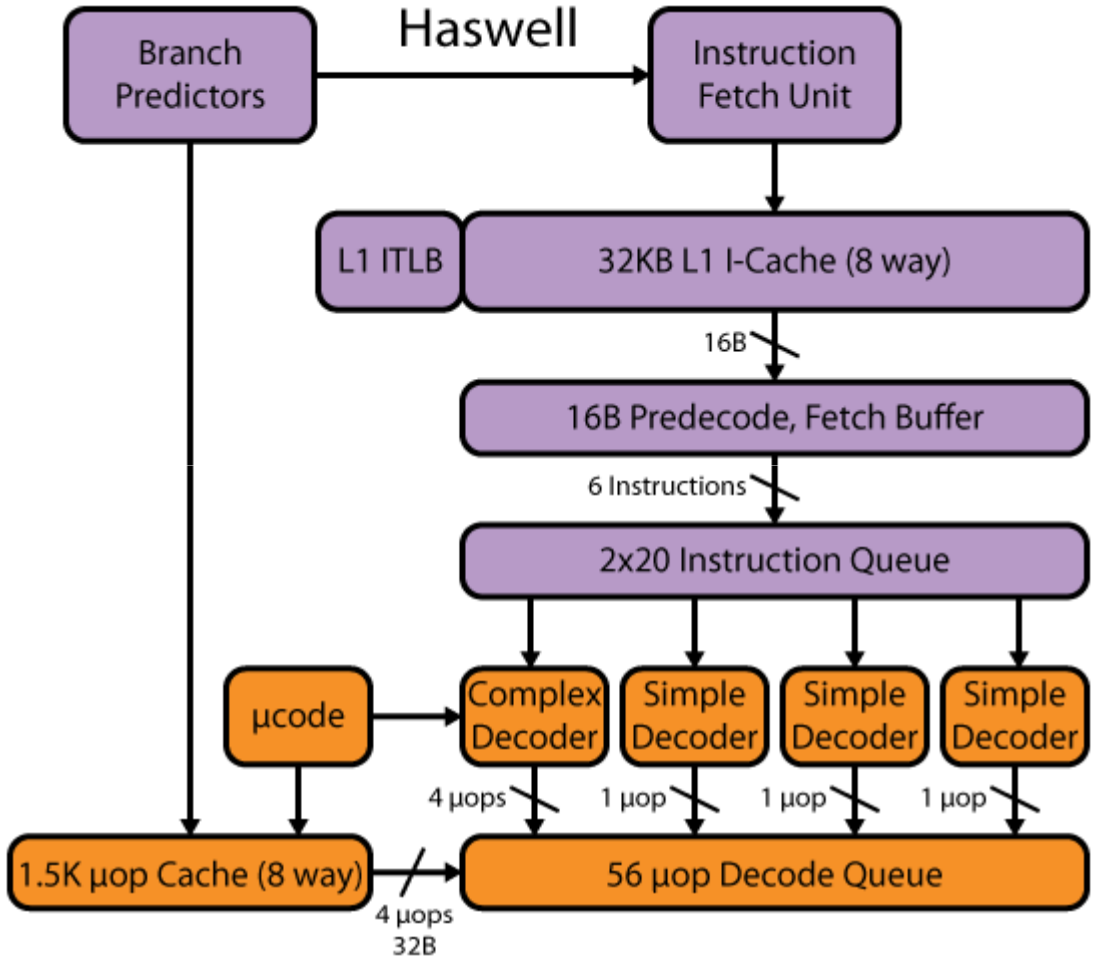
eDRAM (embedded DRAM)



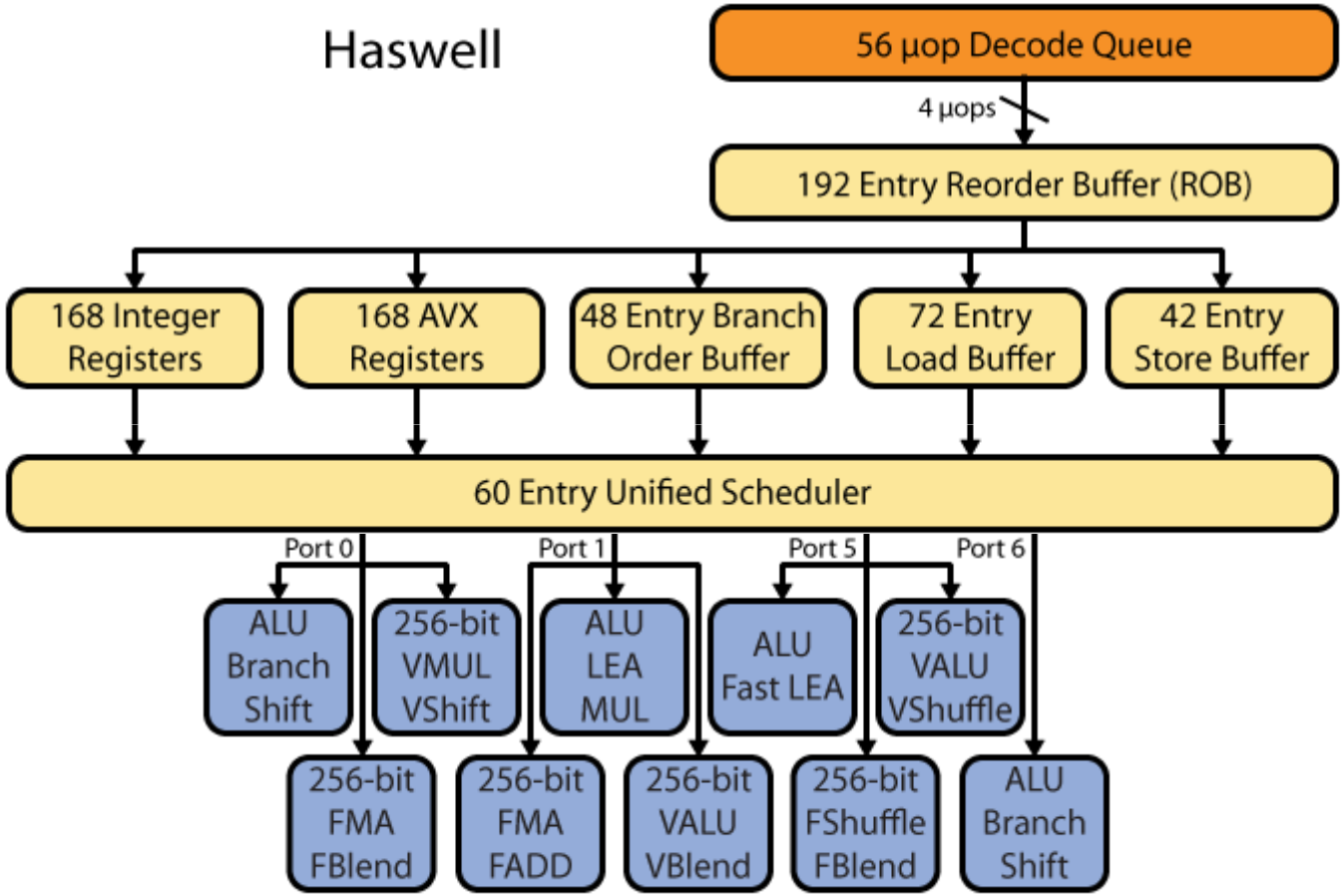
http://www.hotchips.org/wp-content/uploads/hc_archives/hc25/HC25.80-Processors2-epub/HC25.27.820-Haswell-Hammarlund-Intel.pdf

Haswell Front-end

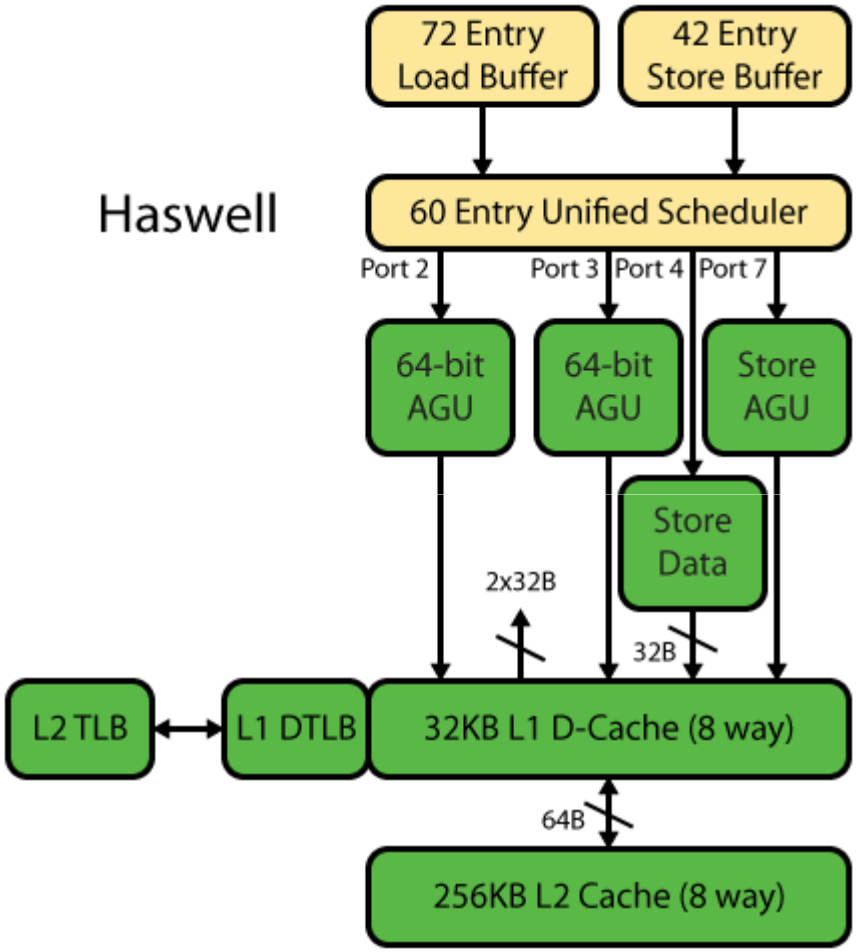
Podobá se Trace cache, ale je jednodušší a instrukce jsou dekódovány



Haswell Out-of-Order Scheduling + Execution Units



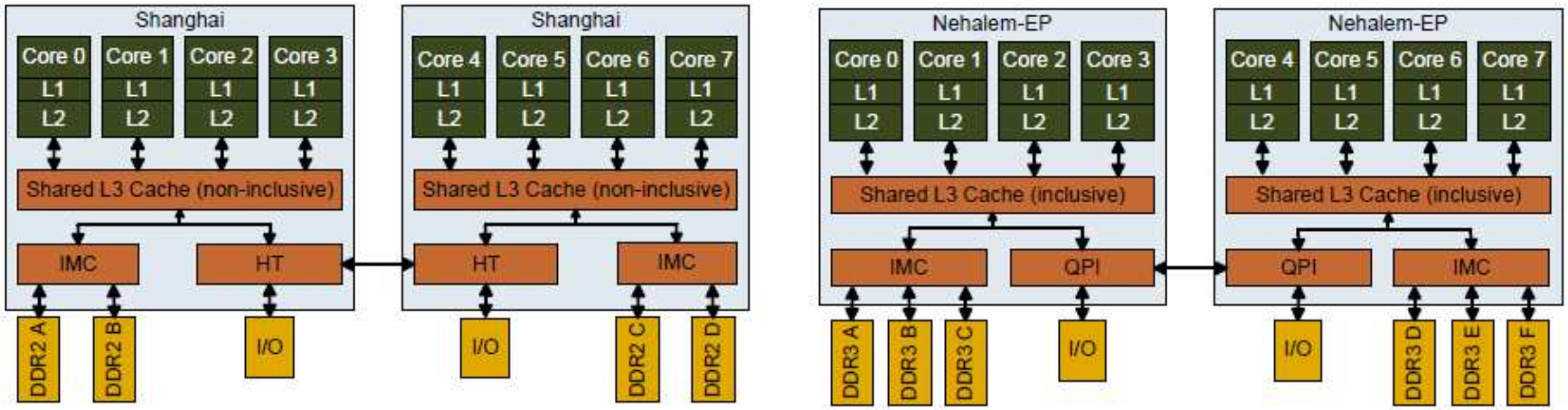
Memory Hierarchy



Porovnání

	Nehalem (2008)	Sandy Bridge (2011)	Haswell (2013)
L1 i-cache	32 KB, 4-way	32 KB, 8-way	32 KB, 8-way
Předdekódování dodává …do Instruction Queue	6 instrukcí	6 instrukcí	6 instrukcí
Počet dekóderů	3+1 complex	3+1 complex	3+1 complex
Počet držených mikro-op v DecodeQ	28	2x28	56
Položek v ROB (out-of-order window)	128	168	192
Položek v Unifikované rezervační stanici / scheduleru	36	54	60
In-flight loads	48	64	72
In-flight stores	32	36	42

AMD (Shanghai) vs. Intel (Nehalem)



Zdroje:

1. Thomadakis, M.E., The Architecture of the Nehalem Processor and Nehalem-EP SMP Platforms, Texas A&M University, 2011.
2. Nehalem (microarchitecture), http://en.wikipedia.org/wiki/Nehalem_%28microarchitecture%29
3. Trent Rolf, Cache Organization and Memory Management of the Intel Nehalem Computer Architecture, University of Utah Computer Engineering, Dec. 2009
4. Paul G. Howard, Next Generation Intel Microarchitecture Nehalem, Microway Inc., 2009
5. Intel QuickPath Interconnect, http://en.wikipedia.org/wiki/Intel_QuickPath_Interconnect
6. http://chip-architect.com/news/Nehalem_at_1st_glance_.jpg
7. http://www.realworldtech.com/haswell-cpu/