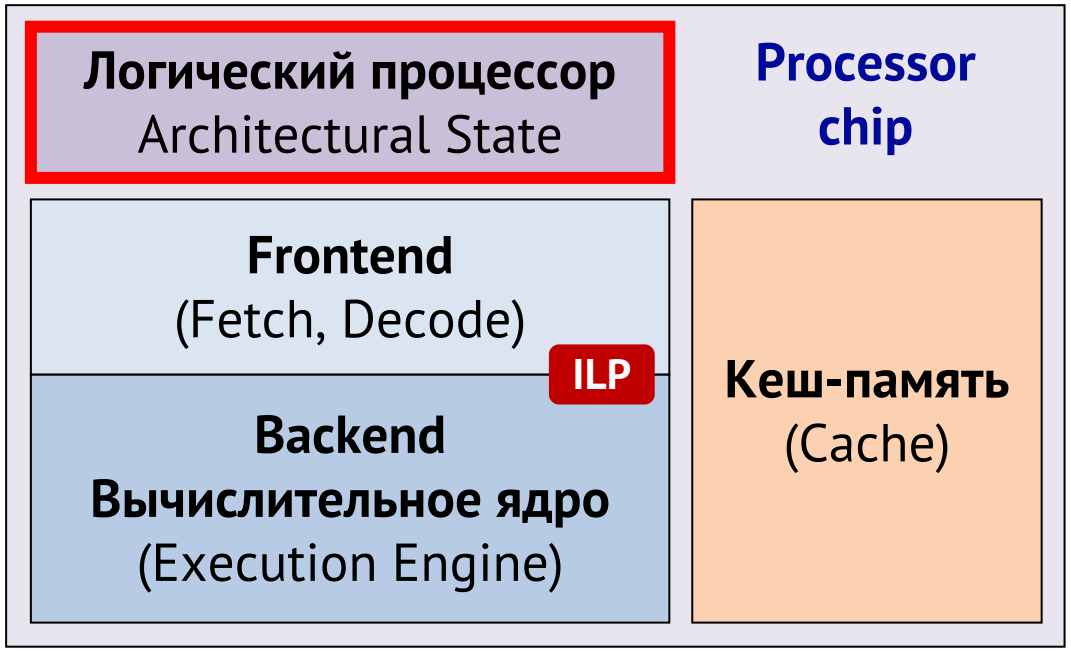
Курс «Архитектурно-ориентированная оптимизация кода»

Структурная организация процессоров с архитектурой Intel 64

Михаил Курносов

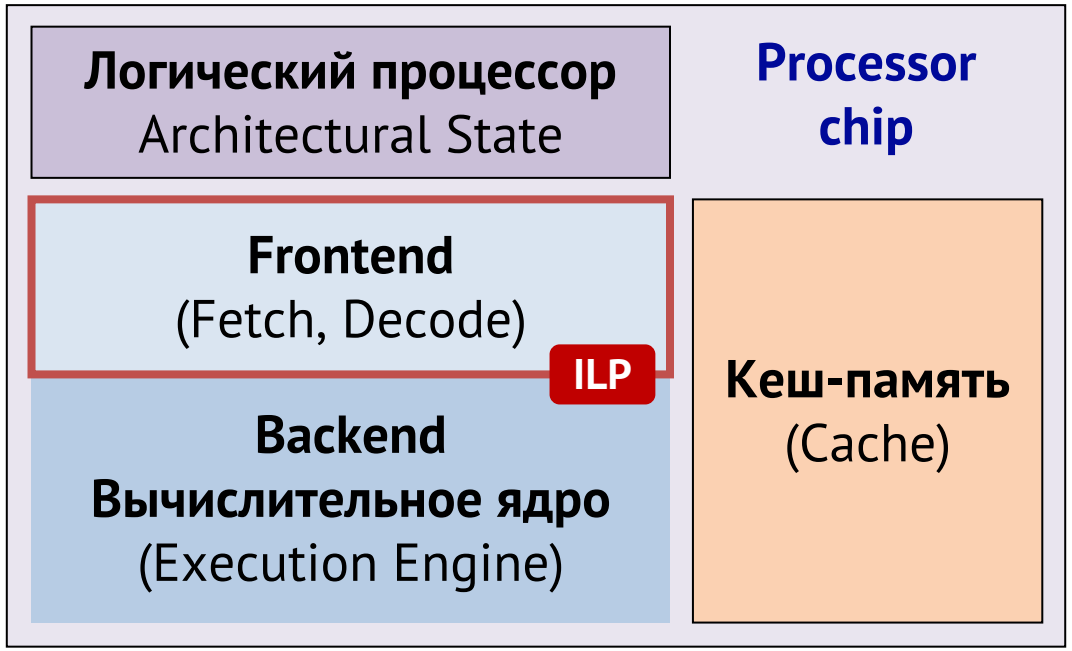
Организация ядра процессора с архитектурой Intel 64



- **Логический процессор (Logical processor)** представлен *архитектурным состоянием* и контроллером прерываний (Interrupt controller, APIC)
- **Архитектурное состояние (Architectural state, AS)** включает:
  - регистры общего назначения (RAX, RBX, …)
  - сегментные регистры (CS, DS, …),
  - управляющие регистры (RFLAGS, RIP, GDTR, …)
  - X87 FPU-регистры, MMX/XMM/YMM-регистры
  - MSR-регистры (time stamp counter, …)
- **Логический процессор** – это то, что “видит” операционная система

- Intel64 and IA-32 Architectures Software Developer Manuals // https://software.intel.com/en-us/articles/intel-sdm

Организация ядра процессора с архитектурой Intel 64



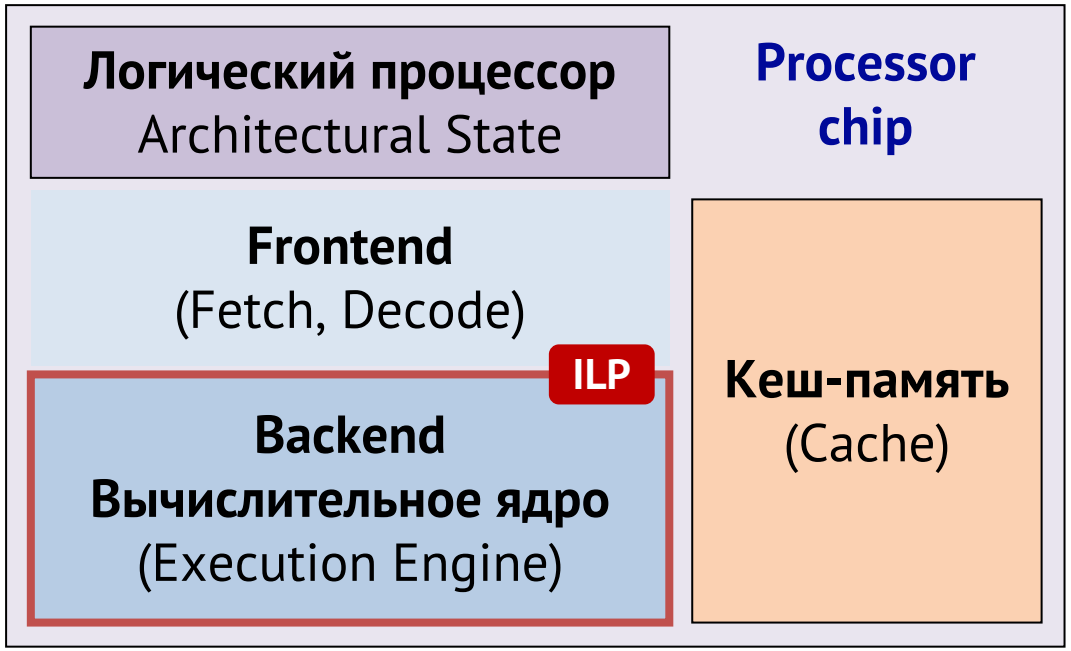
- **Логический процессор** использует ресурсы вычислительного ядра

Frontend

- *загружает* (fetch) из кеш-памяти/памяти CISC инстуркций Intel 64
- *декодирует* (decode) CISC инструкций Intel 64, разбивает их на RISC микрооперации (uops) и помещает в *очередь* для Backend

- Frontend включает модуль предсказания переходов (branch prediction unit), который на основе истории ветвлений возвращает адрес условного перехода
- Frontend реализует спекулятивную выборку и декодирование инструкций

Организация ядра процессора с архитектурой Intel 64

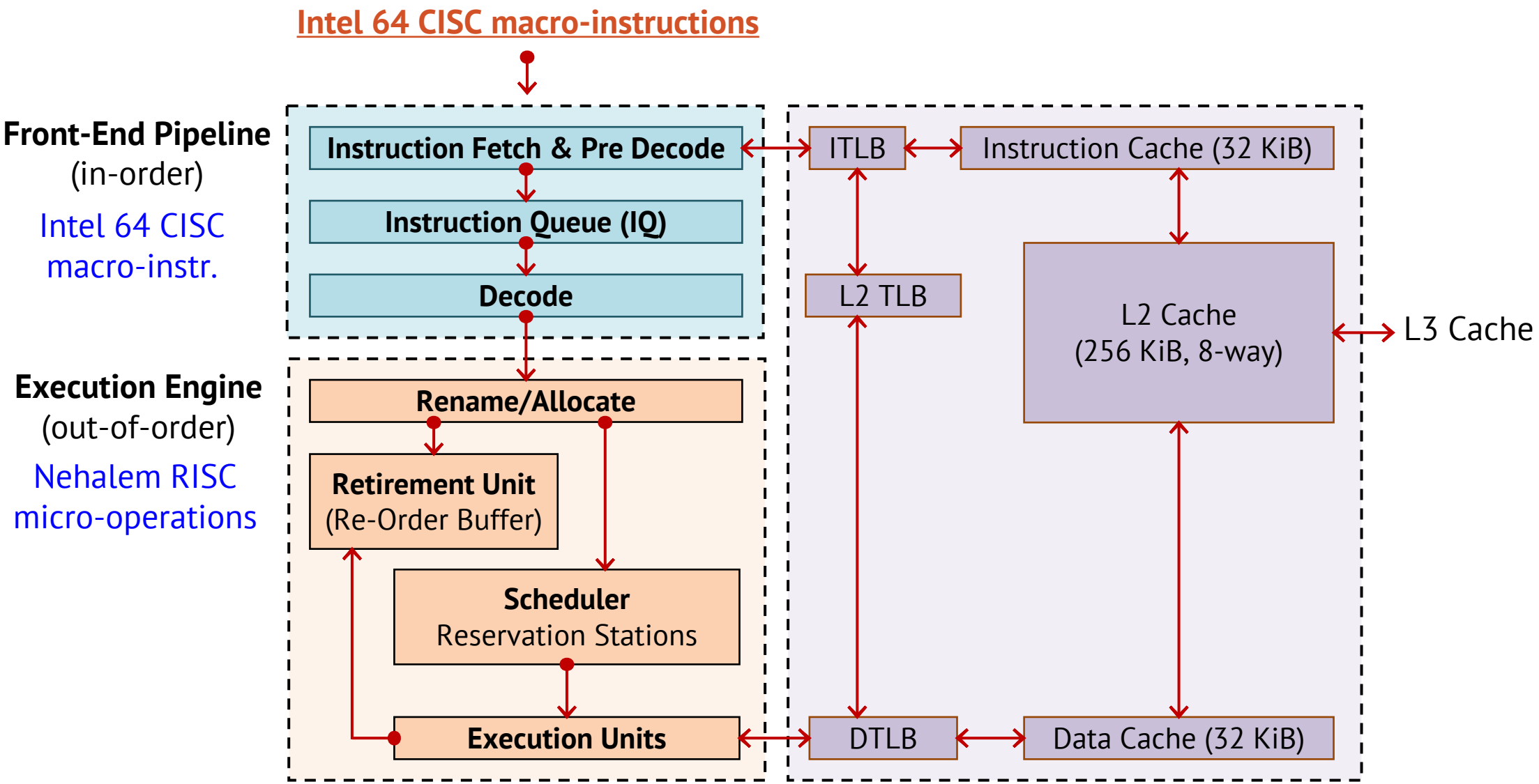


Backend

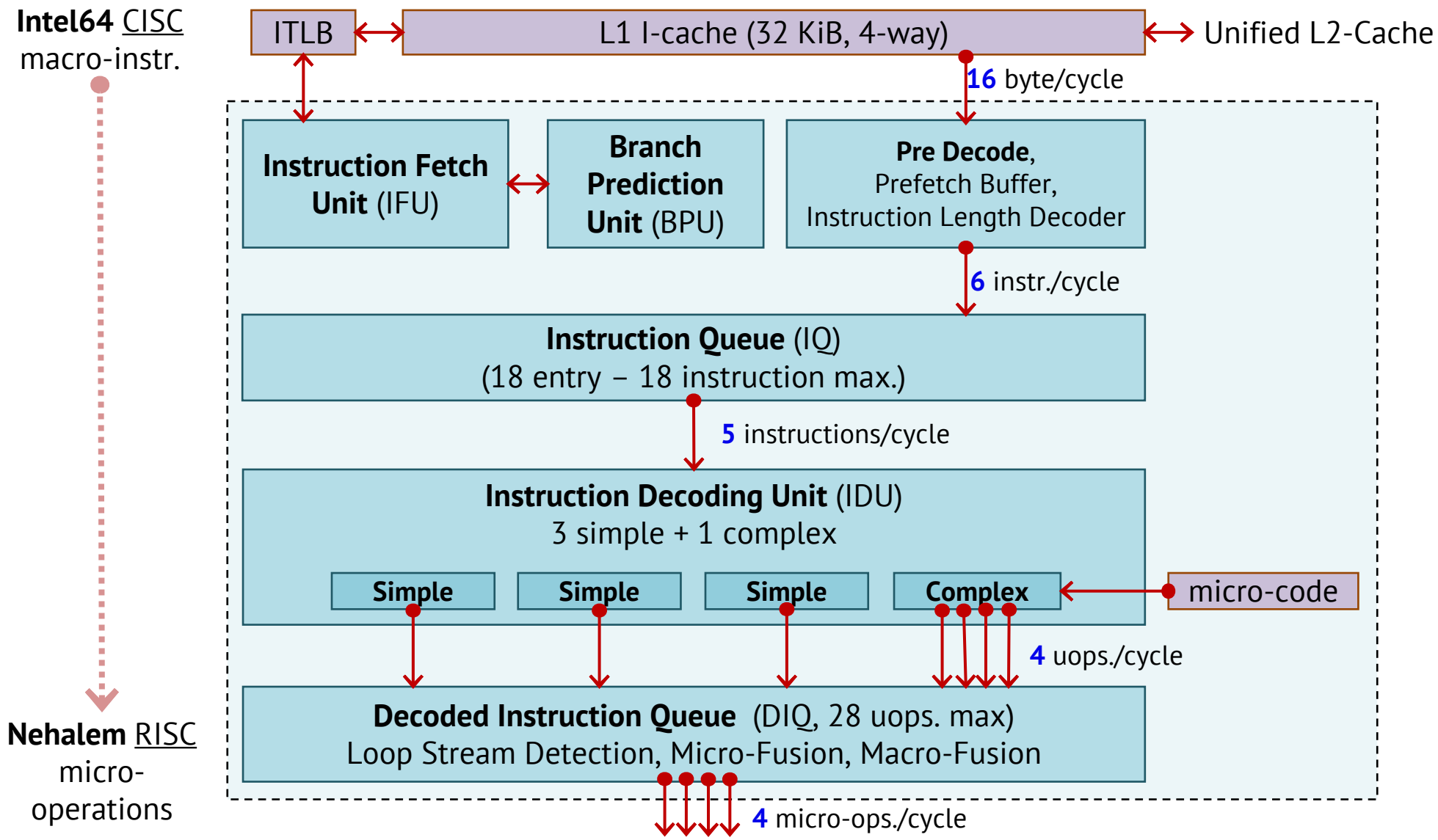
- *планирует* внеочередное параллельное выполнение RISC микроопераций на доступных модулях (портах: ALU, FPU, LD/ST)
- реализует внеочредное параллельное *выполнение* микроопераций по готовности операндов (out-of-order execution)
- *записывает* (retire) результат операции в архитектурное состояние/память в исходном порядке программы (in-order)

- Backend реализует параллелизм уровня инструкций (Instruction Level Parallelism): параллельное выполнение микроопераций на доступных функциональных модулях, внеочередное выполнение, SIMD-инструкции

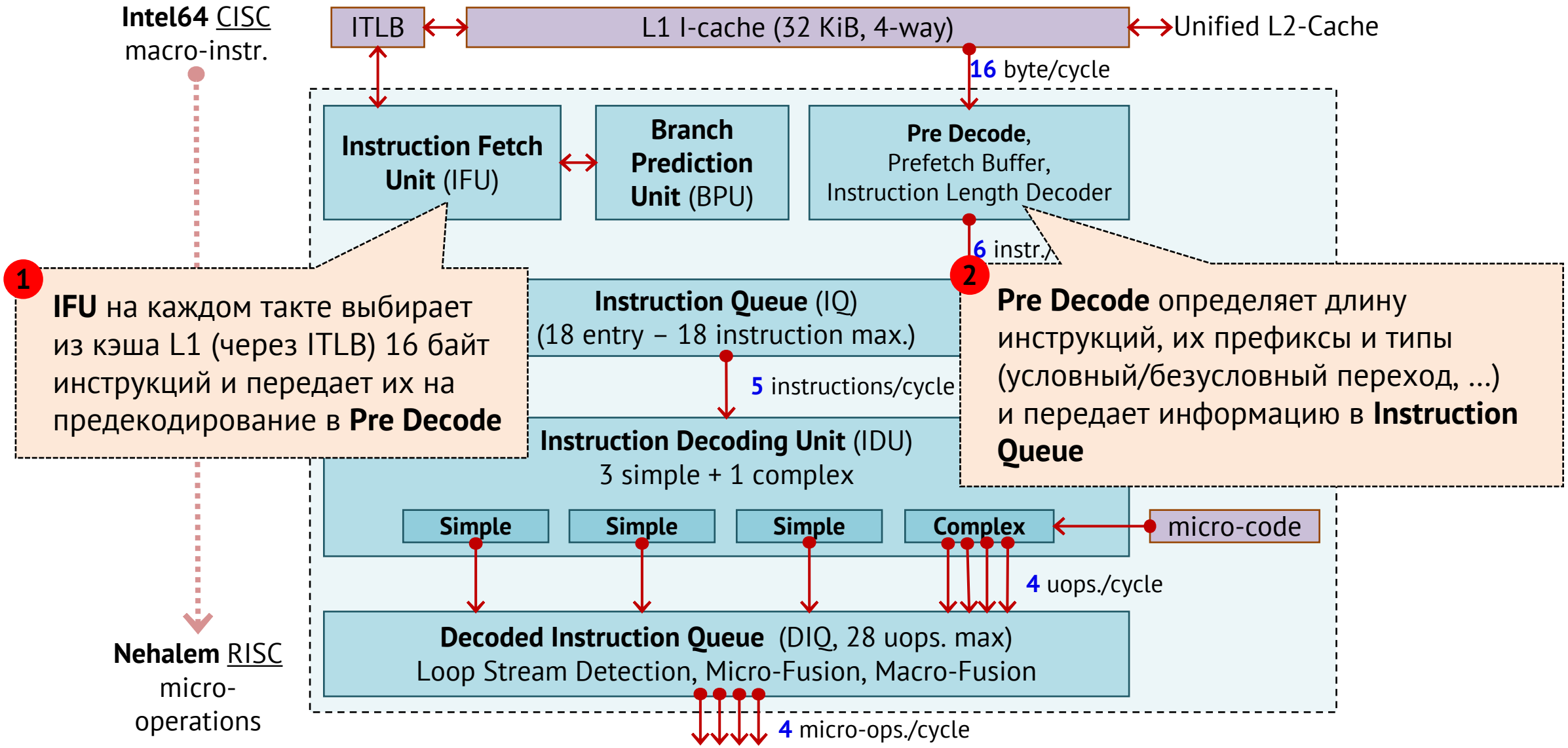
Intel Nehalem Core Pipeline



Intel Nehalem Frontend Pipeline (in-order)



Intel Nehalem Frontend Pipeline (in-order)

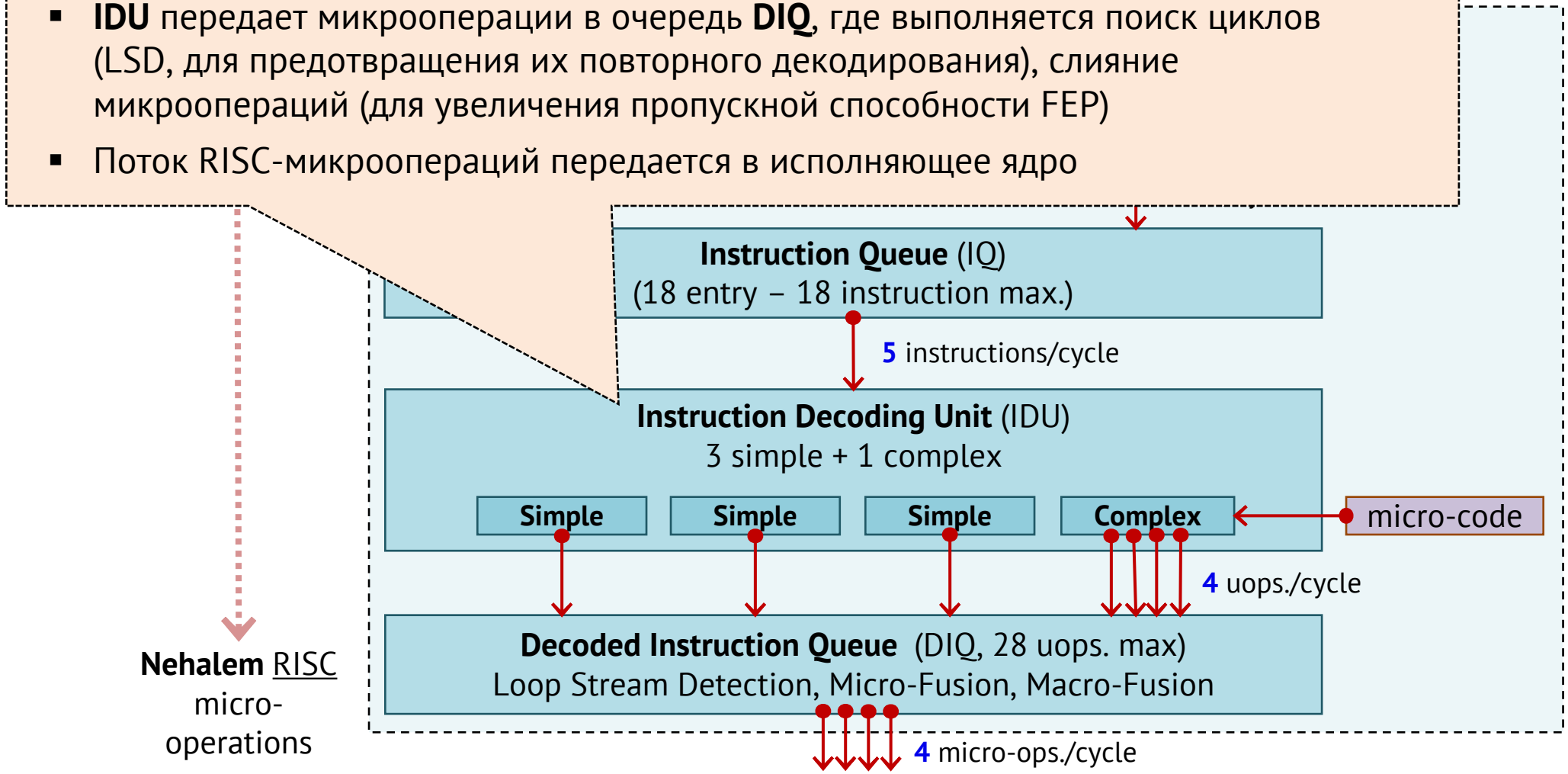


Intel Nehalem Frontend Pipeline (in-order)

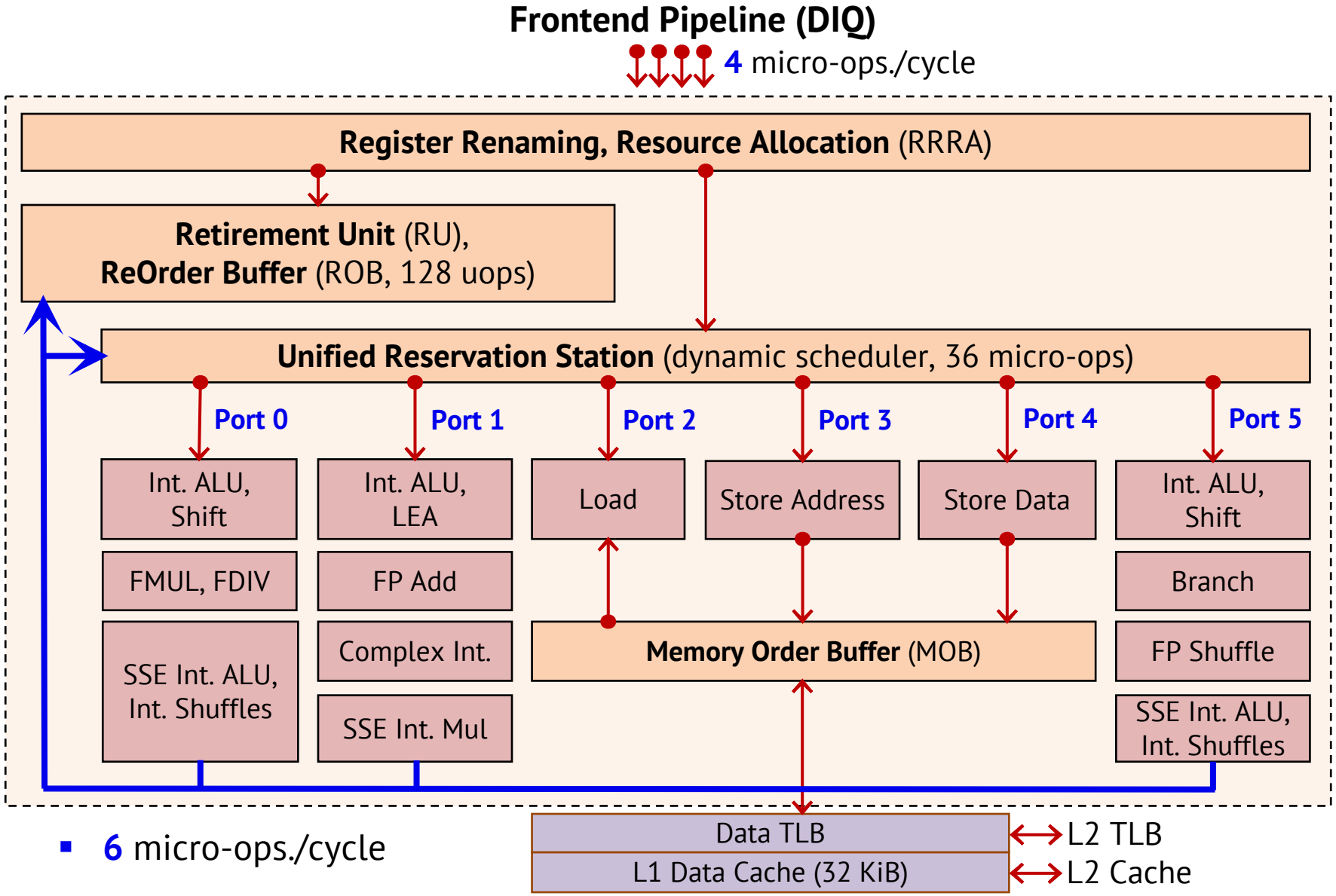
3

- **IDU** преобразует Intel64 инструкции в RISC-микрооперации (сложные инструкции преобразуются в несколько микроопераций)
- **IDU** передает микрооперации в очередь **DIQ**, где выполняется поиск циклов (LSD, для предотвращения их повторного декодирования), слияние микроопераций (для увеличения пропускной способности FEP)
- Поток RISC-микроопераций передается в исполняющее ядро

d L2-Cache

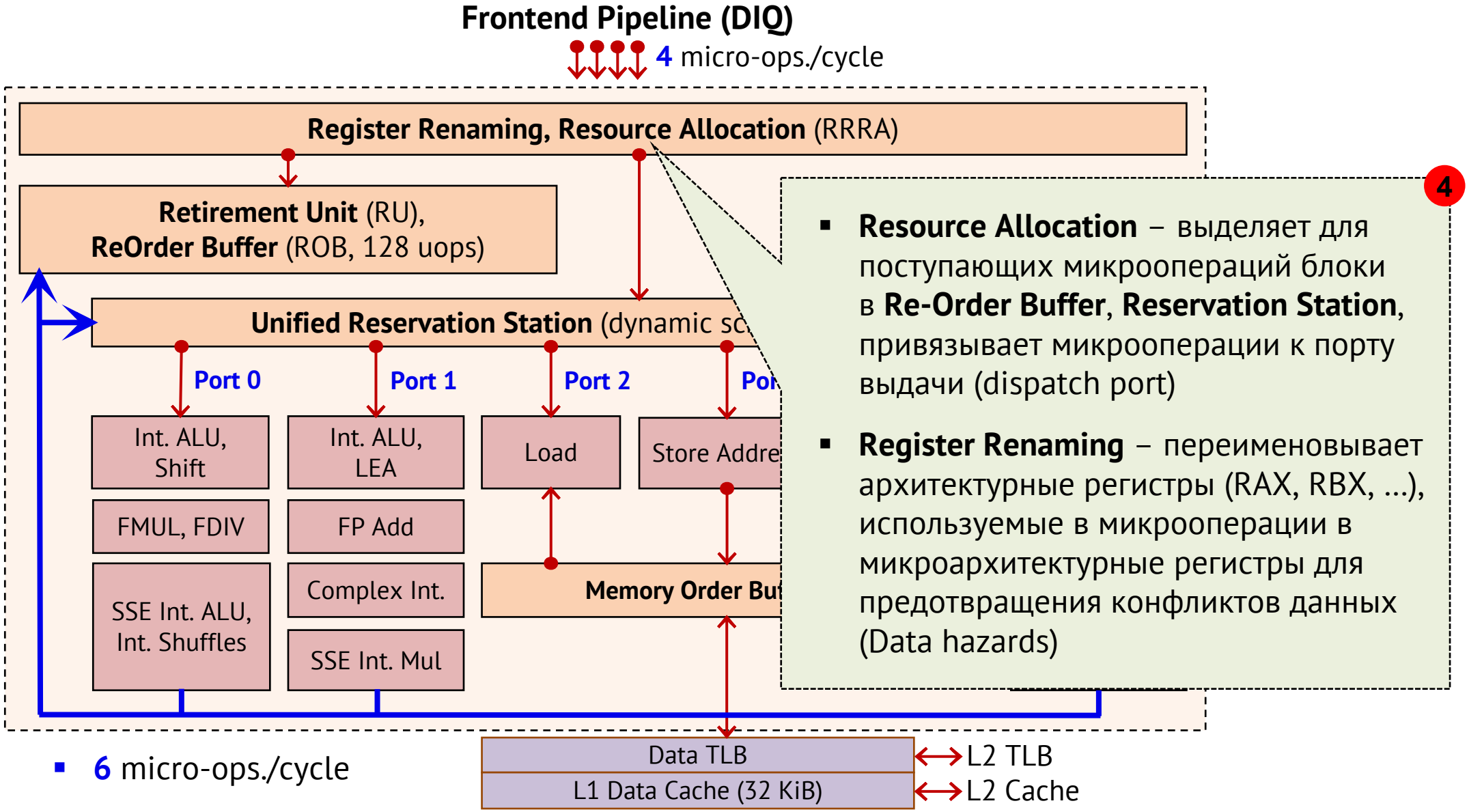


Intel Nehalem Execution Engine

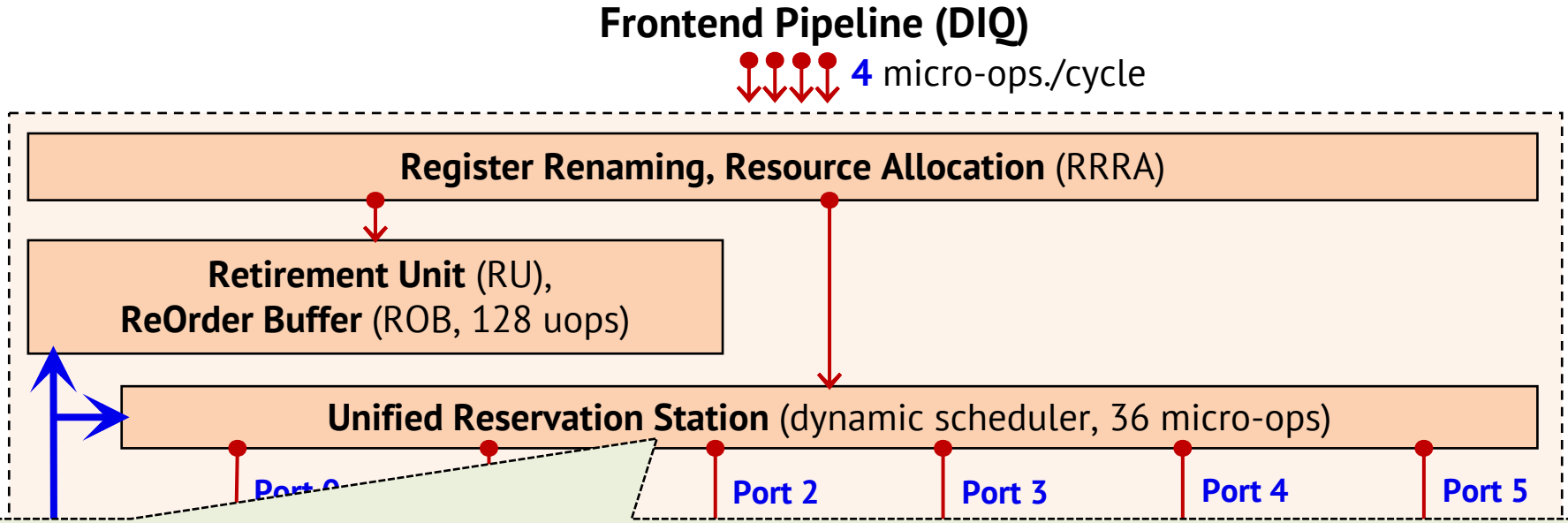


- 6 micro-ops./cycle

Intel Nehalem Execution Engine

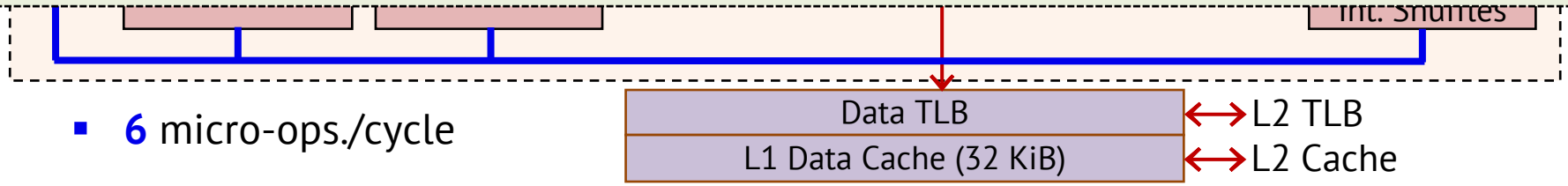


Intel Nehalem Execution Engine



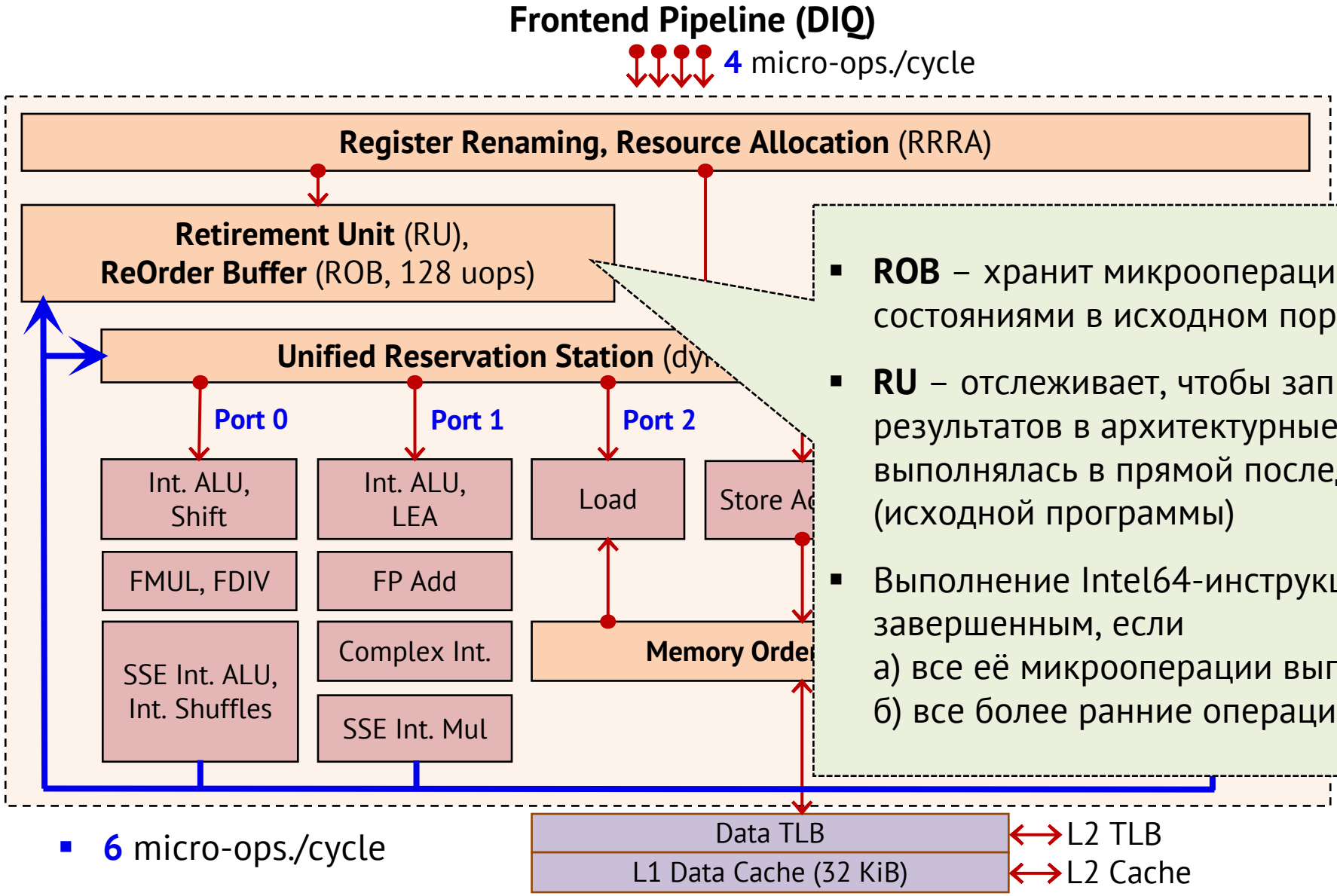
5

- **URS** – пул из 36 микроопераций + динамический планировщик
- Если операнды микрооперации готовы, она направляется на одно их исполняющих устройств – выполнение по готовности данных (максимум 6 микроопераций/такт – 6 портов)
- URS реализует разрешения некоторых конфликтов данных – передает результат выполненной операции напрямую на вход другой (если требуется, forwarding, bypass)



- **6** micro-ops./cycle

Intel Nehalem Execution Engine

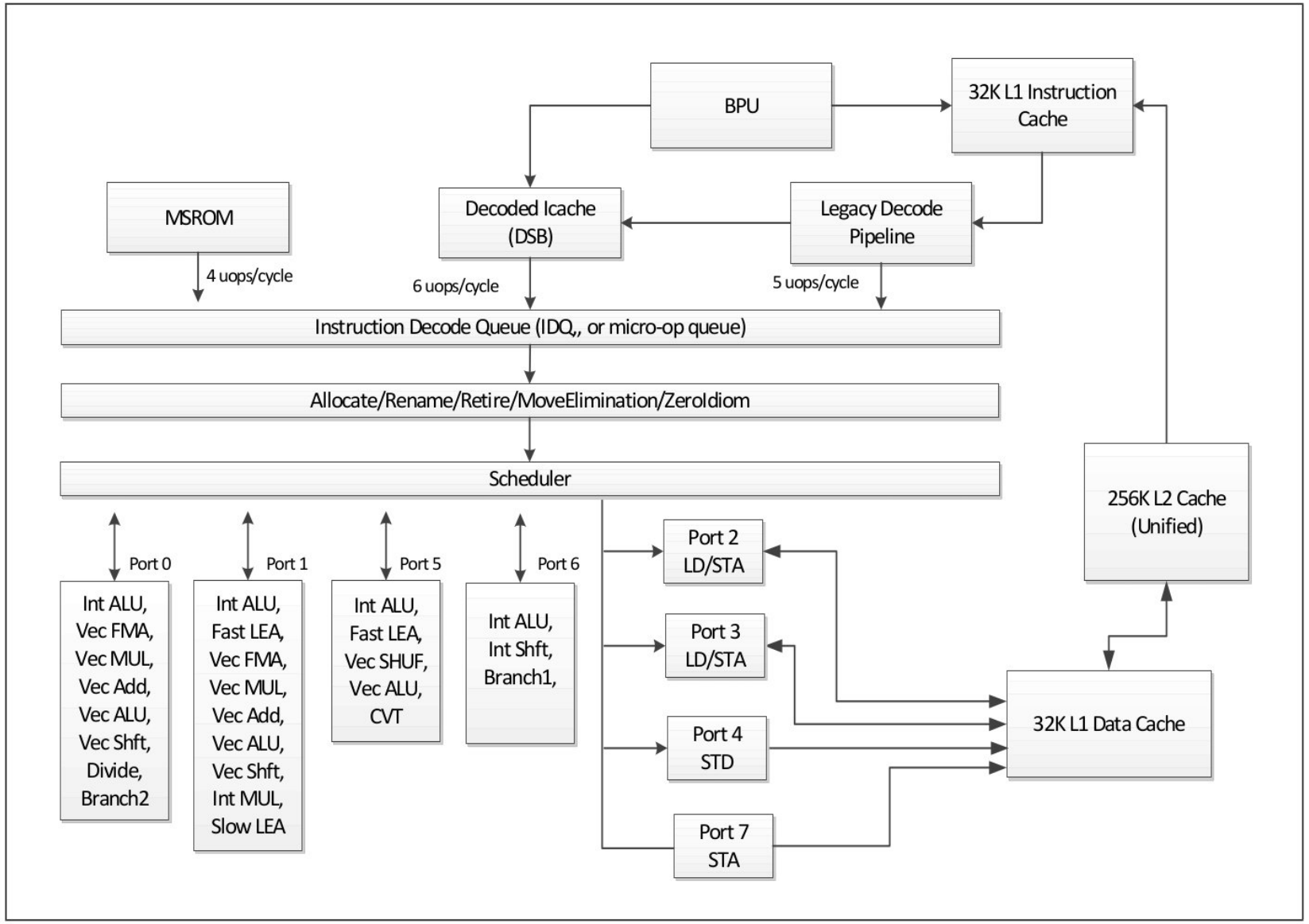


6

- **ROB** – хранит микрооперации с их состояниями в исходном порядке (in-order)
- **RU** – отслеживает, чтобы запись результатов в архитектурные регистры выполнялась в прямой последовательности (исходной программы)
- Выполнение Intel64-инструкции считается завершенным, если
  а) все её микрооперации выполнены,
  б) все более ранние операции завершены

- **6** micro-ops./cycle

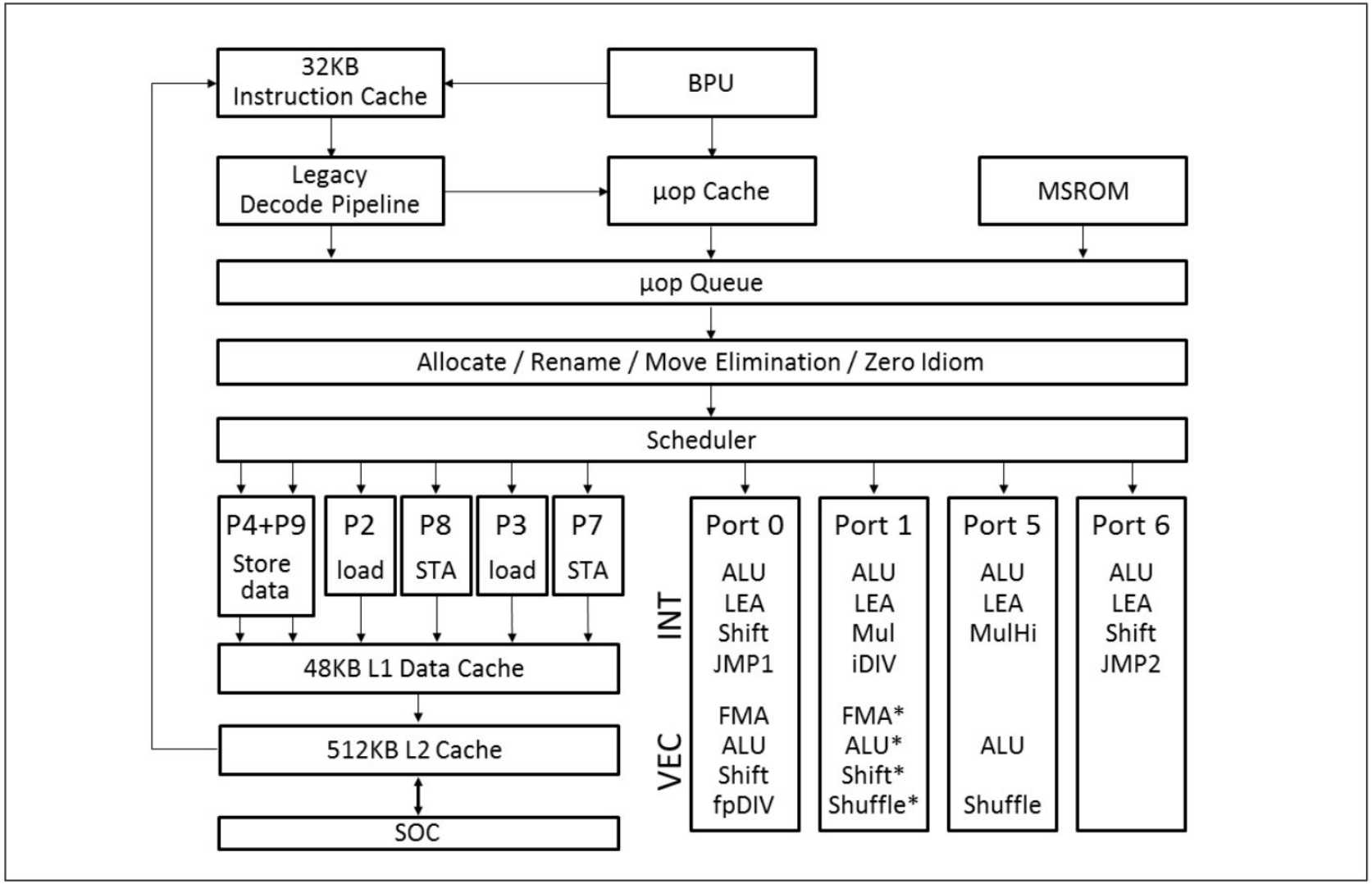
Skylake Microarchitecture



- DSB — 6 uops per cycle
- IDQ — 64 uops per logical processor
- Out-of-Order execution engine
  - Exec. ports: 8

Intel 64 and IA-32 Architectures Optimization Reference Manual

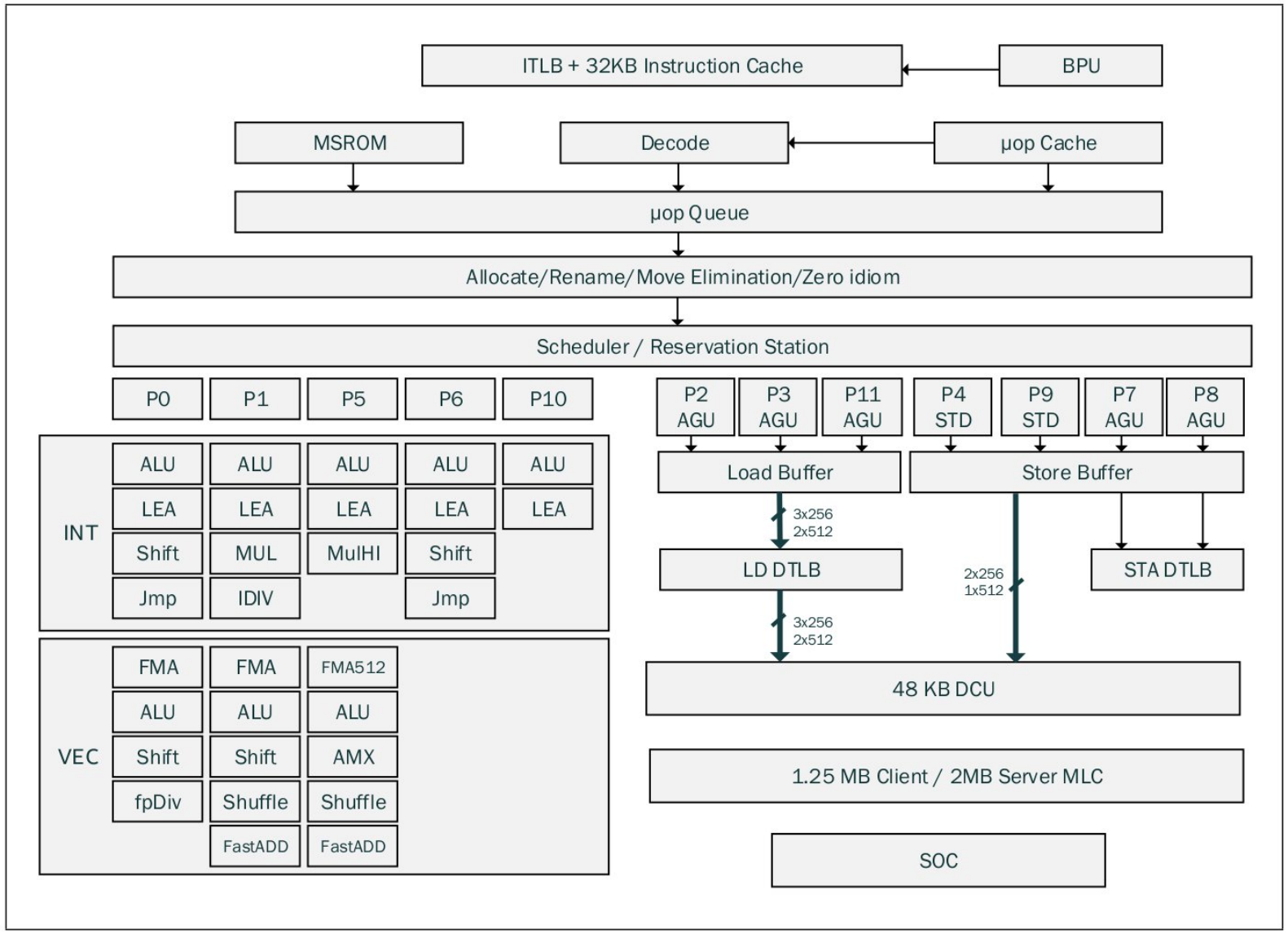
Ice Lake Microarchitecture



- IDQ — 70 uops per logical processor
- Out-of-Order execution engine
  - Allocation: 5 uops per cycle
  - Exec. ports: 10

Intel 64 and IA-32 Architectures Optimization Reference Manual

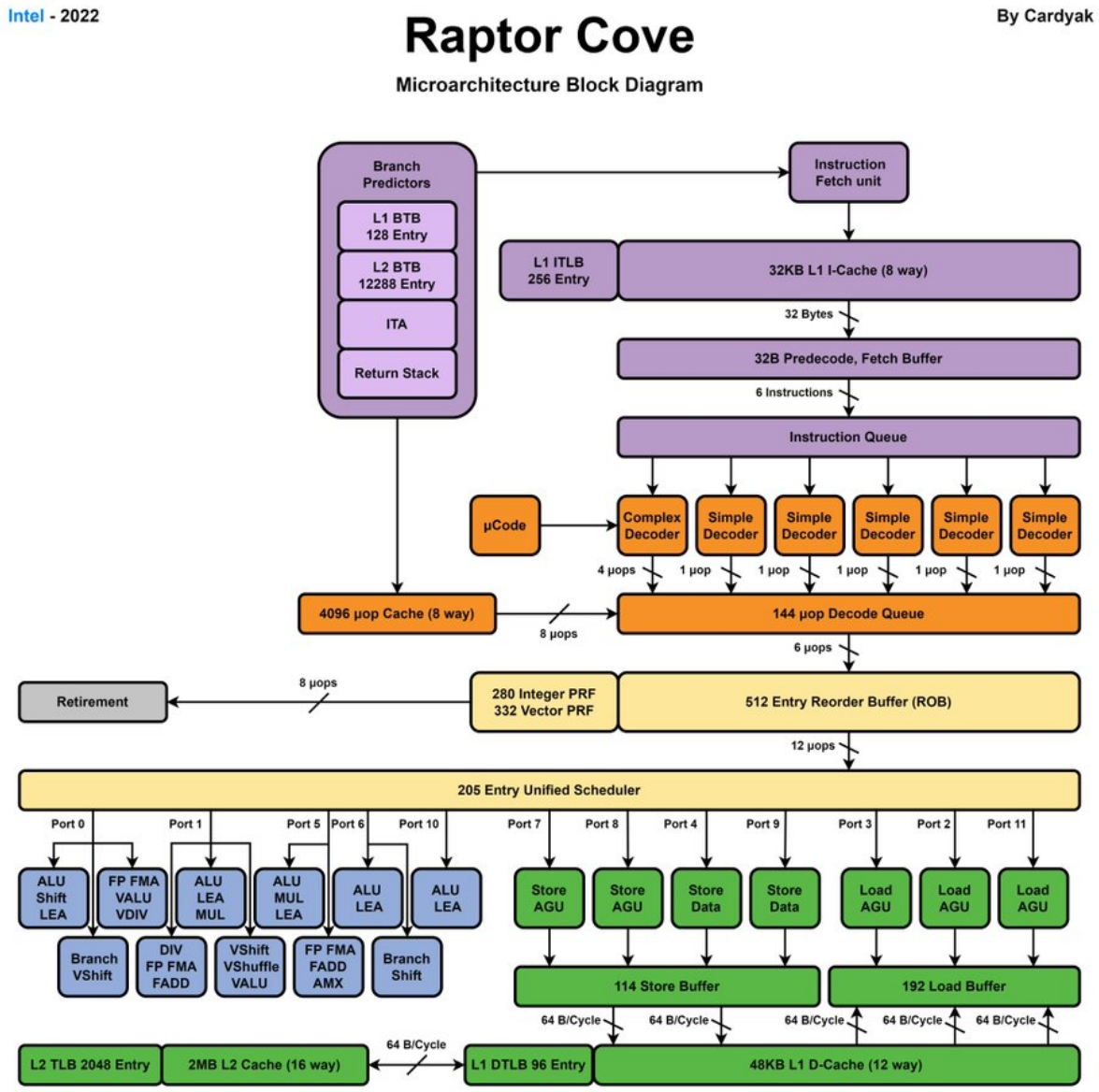
Golden Cove Microarchitecture



- IDQ — 144 uops per logical processor (72 uops per SMT thread)
- Out-of-Order and execution engine
  - Allocation: 6 uops per cycle
  - Retirement: 8 uops per cycle

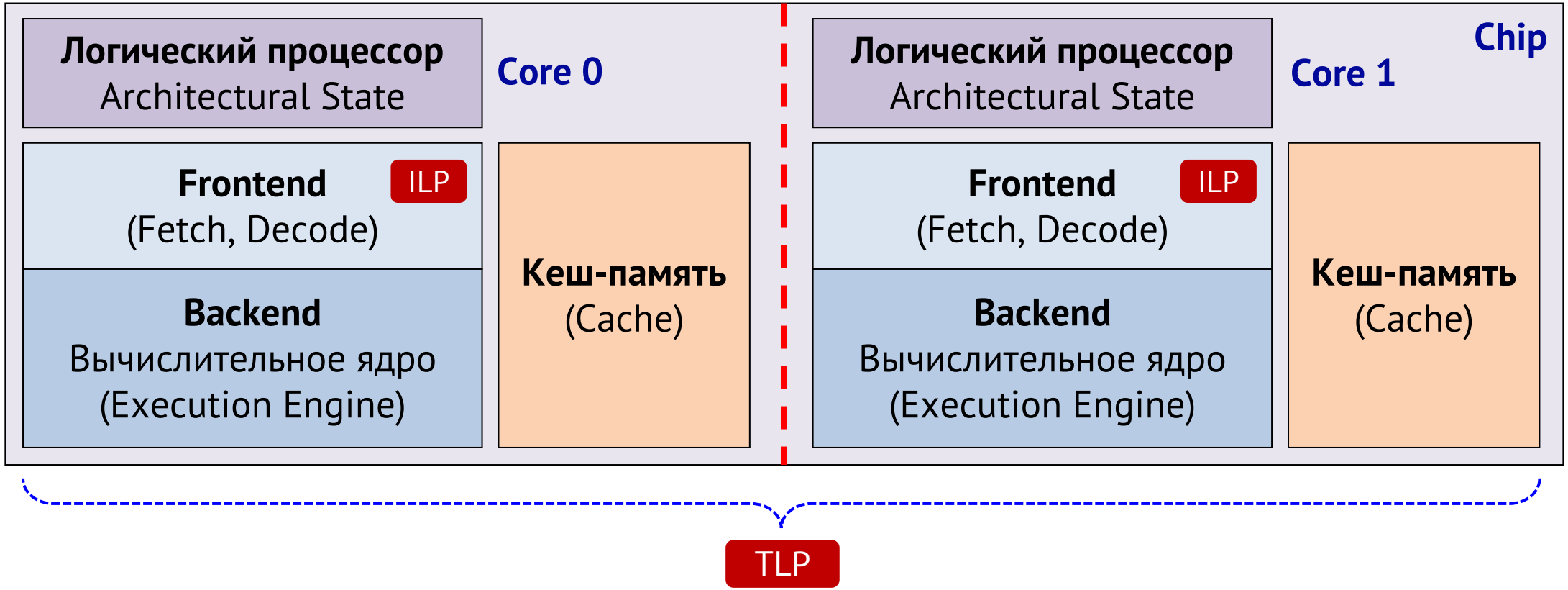
Intel 64 and IA-32 Architectures Optimization Reference Manual

Raptor Cove Microarchitecture (2022)



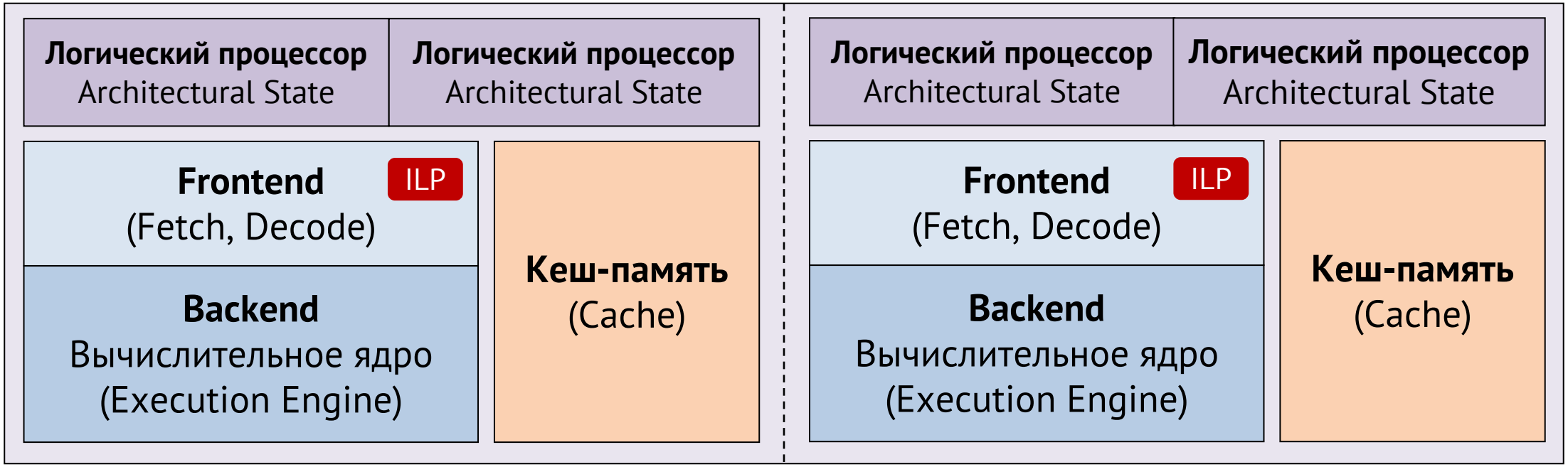
- Boost frequency up to 6.0 GHz
- 2 MB L2 cache per core
- New dynamic prefetcher

Многоядерные процессоры (Multi-core Processors)



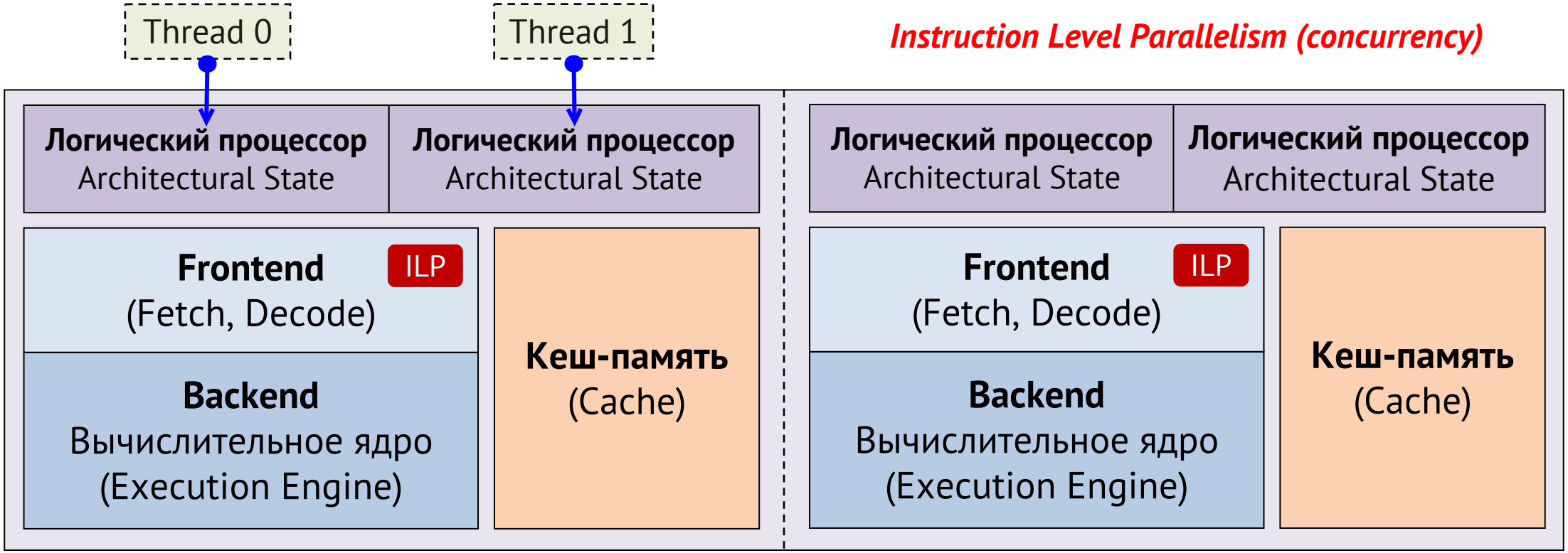
- Ядра процессора могу разделять некоторые ресурсы (например, кеш-память L3)
- Многоядерный процессор реализует параллелизм уровня потоков (Thread Level Parallelism)

Многоядерные процессоры с поддержкой SMT



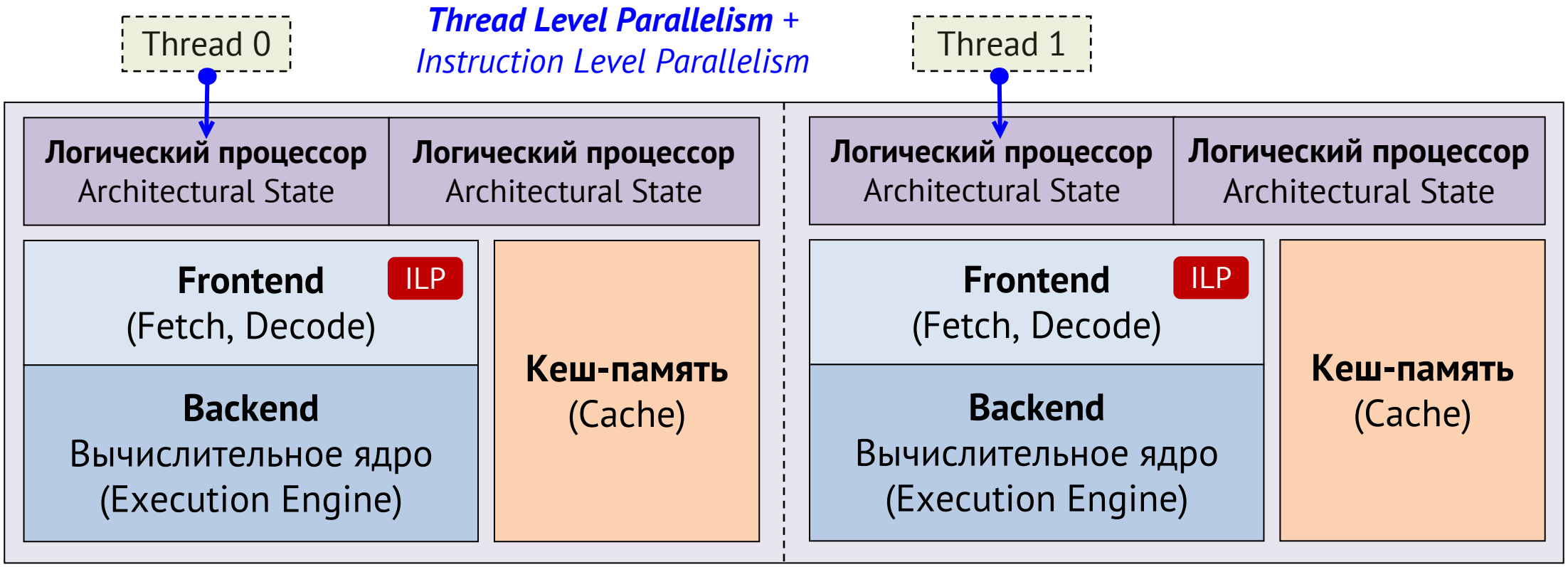
- Аппаратные потоки SMT (hardware threads) разделяют один суперскалярный конвейер ядра (параллелизм инструкций)
- SMT позволяет загрузить конвейер инструкциями разнородных программных потоков
- Simultaneous multithreading (SMT): Intel Hyper-Threading, Fujitsu Vertical Multithreading
- Операционная система представляет каждый SMT-поток как логический процессор

Многоядерные процессоры с поддержкой SMT



- Операционной системе доступны 4 логических процессора (аппаратных потока)
- Программные потоки 0 и 1 выполняются на ресурсах одного ядра – привязаны к логическим процессорам SMT
- Оба потока разделяют ресурсы одного суперскалярного конвейера – конкурируют за ресурсы (только параллелизм уровня инструкций)

Многоядерные процессоры с поддержкой SMT



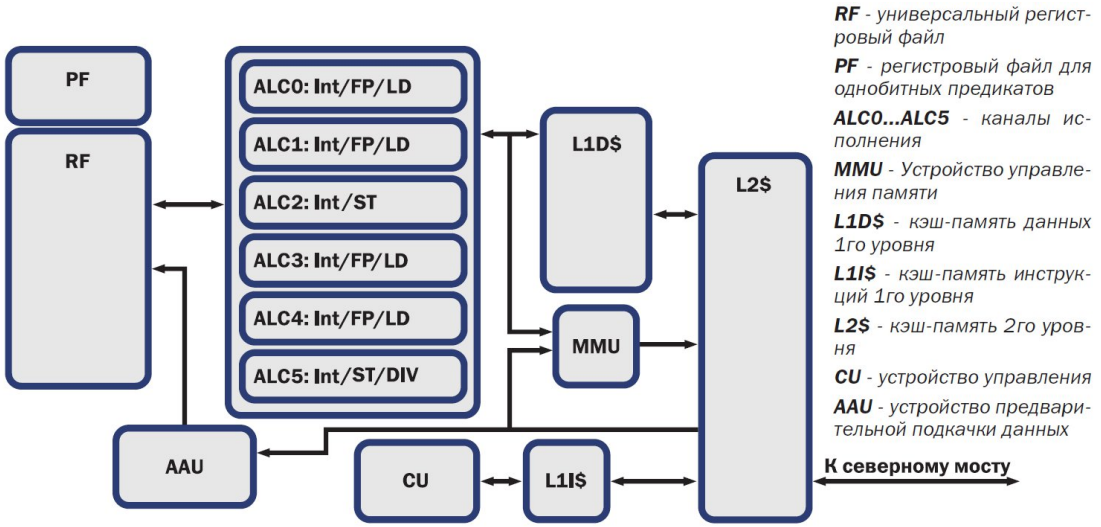
- Операционной системе доступны 4 логических процессора (аппаратных потока)
- Потоки 0 и 1 выполняются на суперскалярных конвейерах разных ядер
- Задействован параллелизм уровня потоков (TLP) и инструкций (ILP)

Микропроцессор «Эльбрус-16С» (1891ВМ038)

- Архитектура: Эльбрус, версия 6
- 16 ядер в процессоре
- 4/2 процессора в модуле

- Кеш-память:
  - L1: 64 Кбайт данные + 128 Кбайт команды в каждом ядре
  - L2: 1 Мбайт в каждом ядре, 16 Мбайт суммарно
  - L3: 32 Мбайт в процессоре

- Память: 4 канала DDR4-2400 ECC



- Широкая команда — в одном такте до 23 операций (> 33 операций при упаковке в векторные команды)
- 6 каналов АЛУ, работающих параллельно
- Регистровый файл: 256 84-разрядных регистров
- Аппаратная предвыборка данных
- Спекулятивное выполнение, поддержка однобитовых предикатов (минимизация числа переходов и параллельное выполнение нескольких ветвей)

http://mcst.ru/elbrus-16s

Альтернативные RISC-архитектуры?

- ARMv8: Apple Mx, Cavium ThunderX, Huawei Kunpeng
- SPARC
- MIPS
- RISC-V
  - https://riscv.org/alliances/
  - https://riscv-alliance.ru/
  - https://cloudbear.ru/
  - https://syntacore.com/
  - SCR1 RISC-V Core (SystemVerilog) // https://github.com/syntacore/scr1