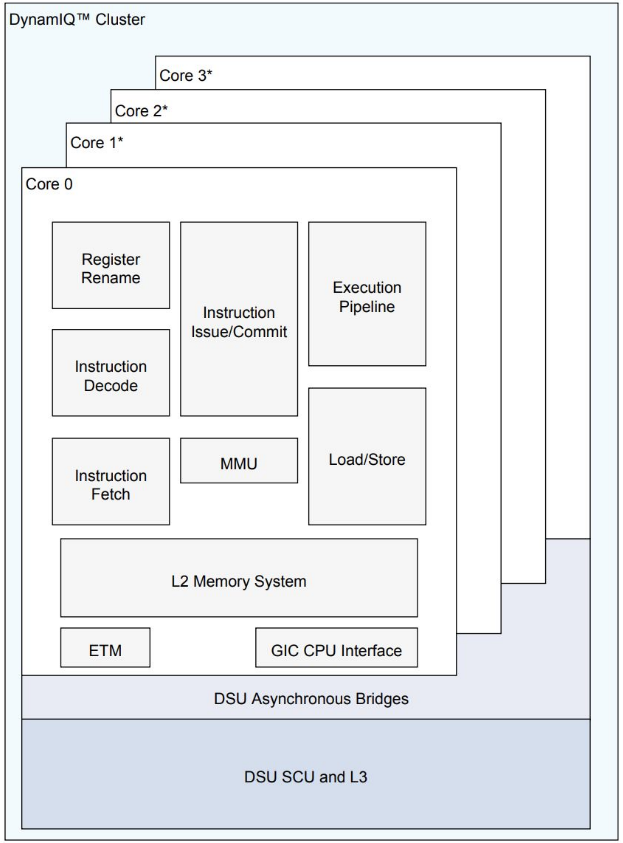
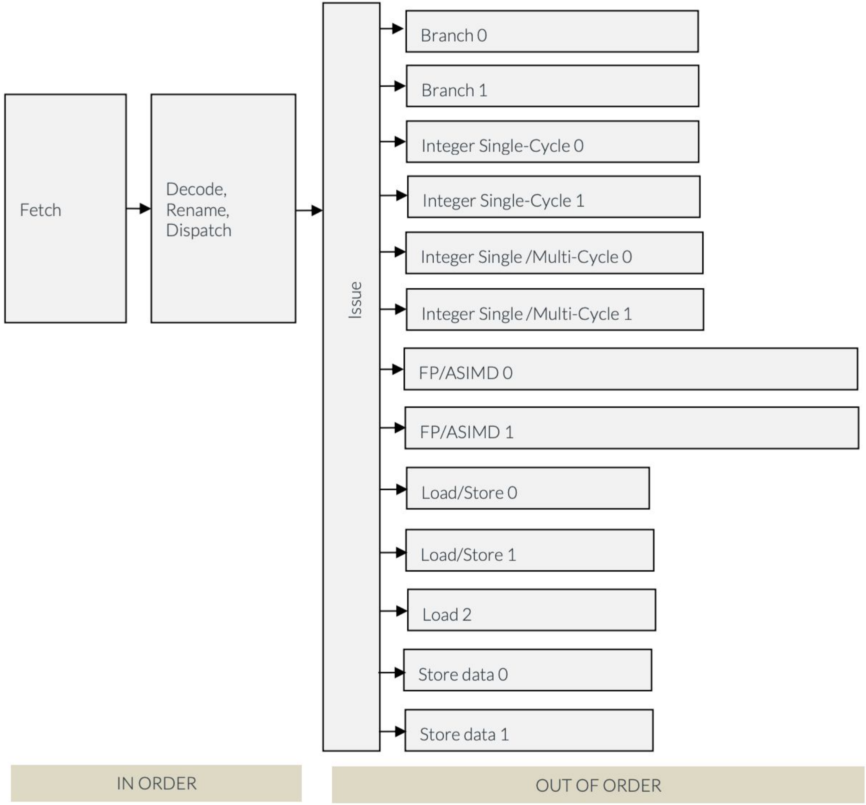
CS 433 Mini-Project: ARM Cortex-A78

Nandeeka Nayak, Jovan Stojkovic,
Antonis Psistakis

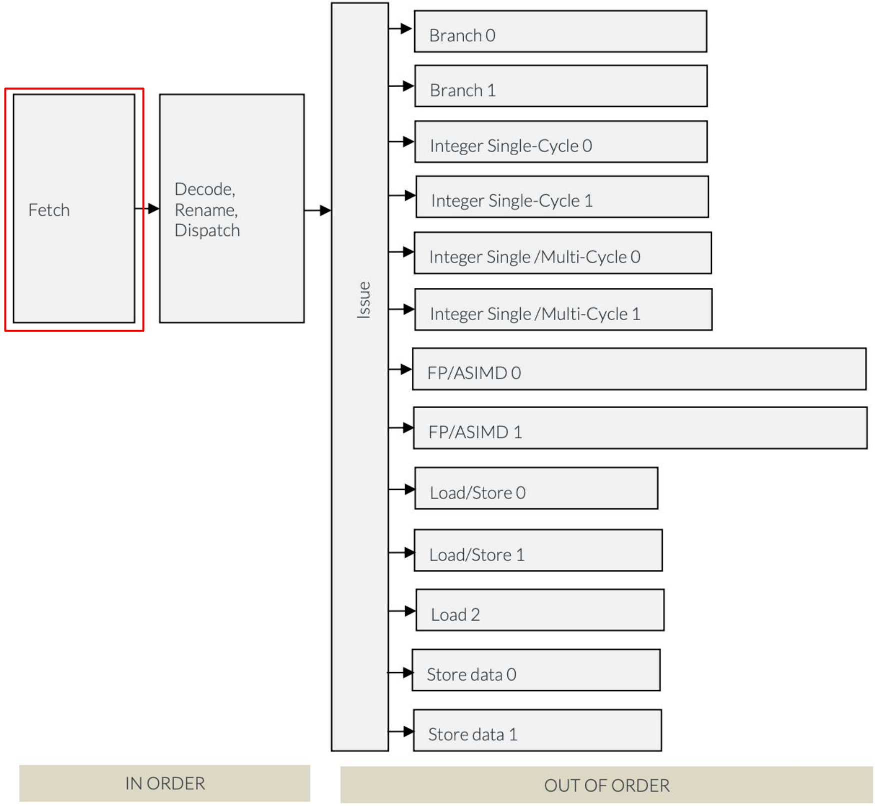
Microarchitecture Overview



Pipeline Overview



Pipeline Overview



Fetch

Decode, Rename, Dispatch

Issue

Branch 0

Branch 1

Integer Single-Cycle 0

Integer Single-Cycle 1

Integer Single /Multi-Cycle 0

Integer Single /Multi-Cycle 1

FP/ASIMD 0

FP/ASIMD 1

Load/Store 0

Load/Store 1

Load 2

Store data 0

Store data 1

IN ORDER

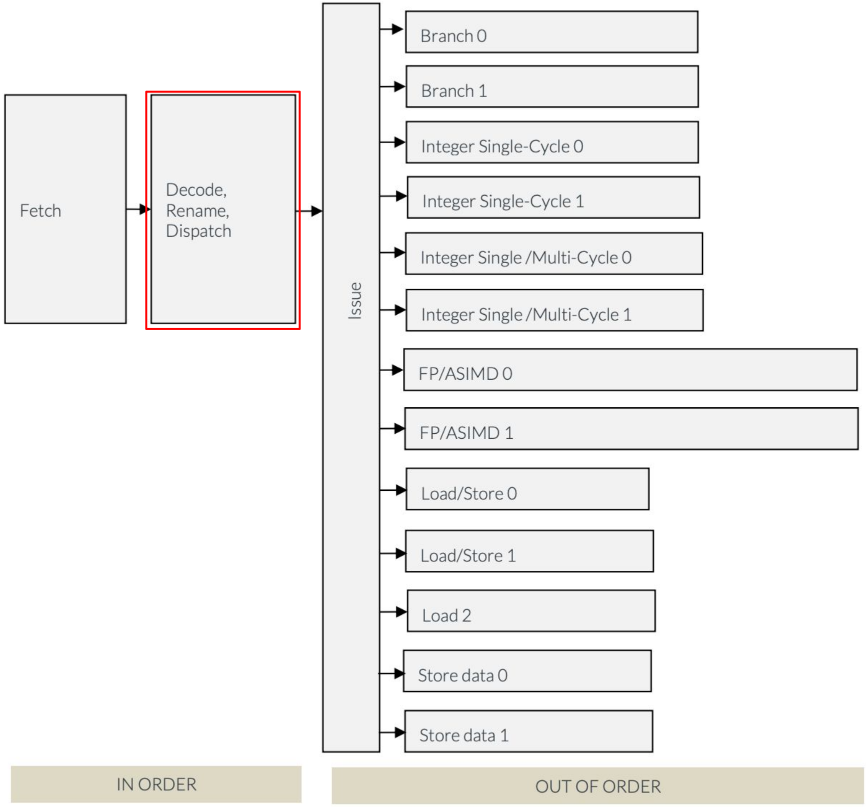
OUT OF ORDER

Instruction Fetch: Branch Prediction

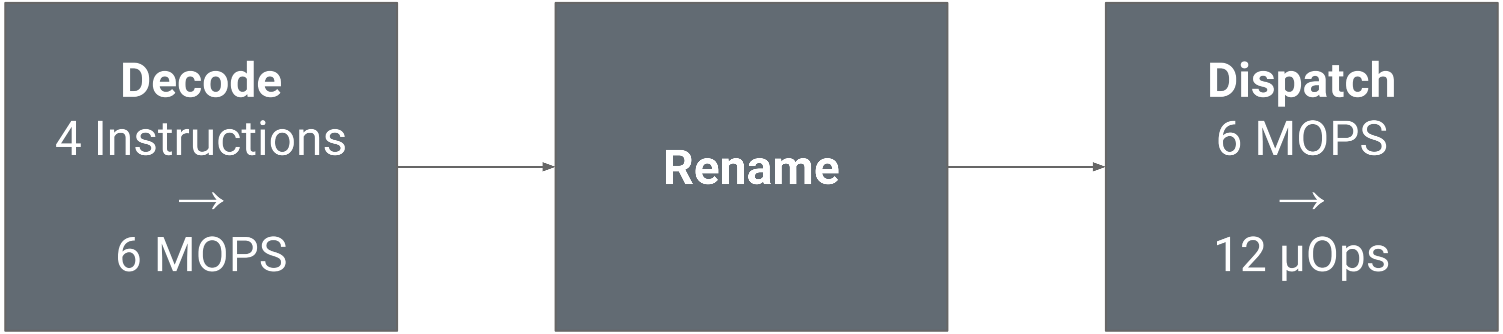
Hardware includes:

- A branch direction predictor using previous branch history
- A static branch predictor
- Branch Target Buffer (BTB)
- The return stack, a stack of nested subroutine return addresses
- An indirect branch predictor

Pipeline Overview



Instruction Decode and Dispatch

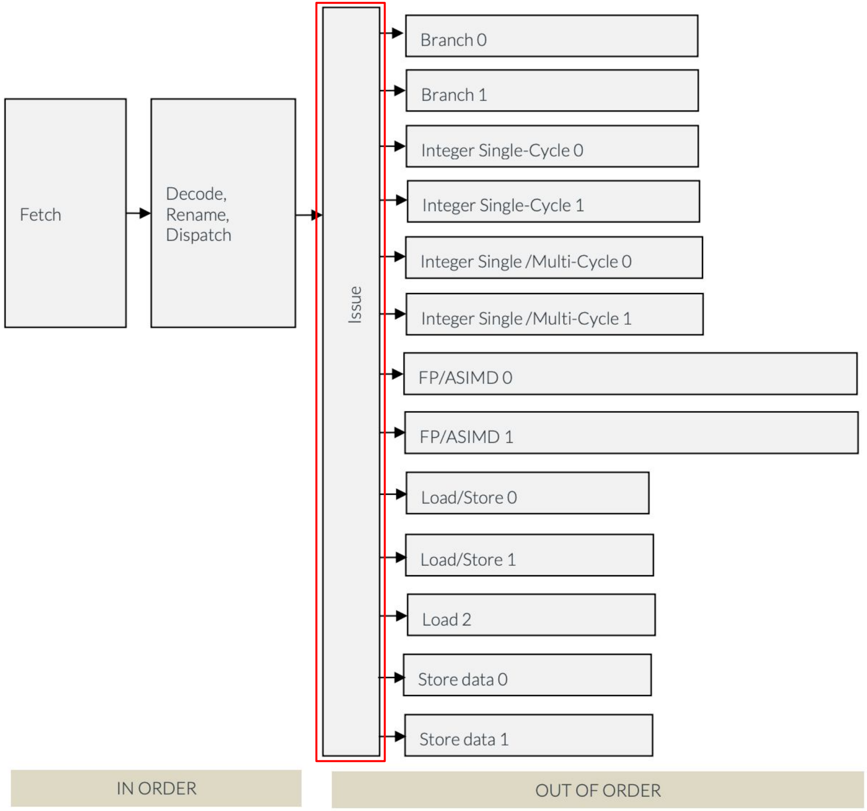


Decode
4 Instructions
→
6 MOPS

Rename

Dispatch
6 MOPS
→
12 µOps

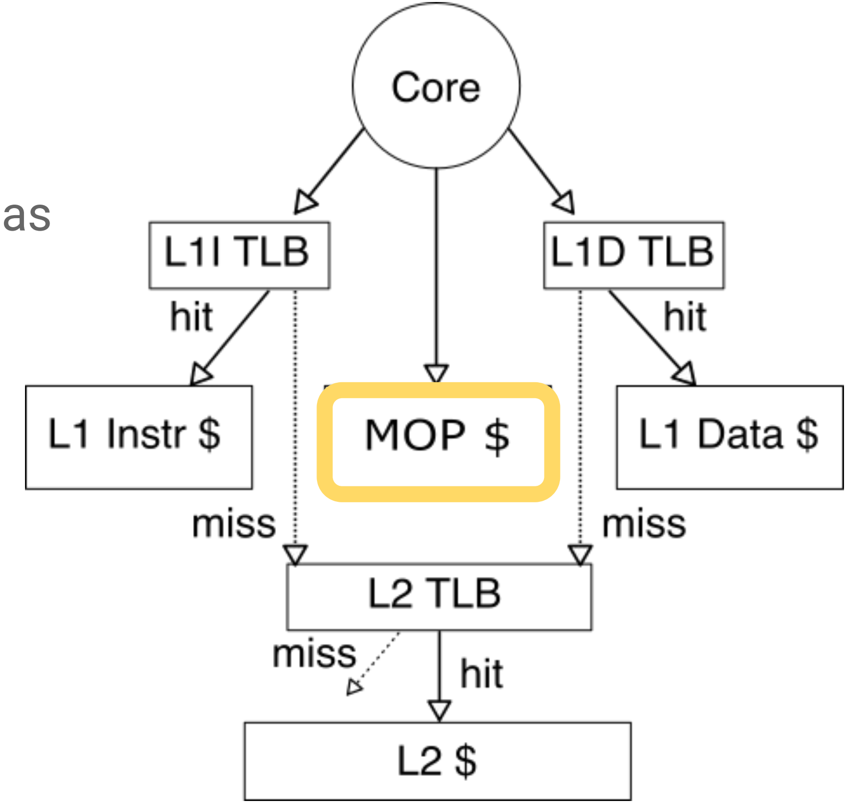
Pipeline Overview



Memory Hierarchy (1/3)

Macro-OP $:

- 1.5K entries
- 4-way Skewed Associative
- Virt. Indexed Virt. Tagged (VIVT) behaves as Phys. Indexed Phys. Tagged (PIPT)



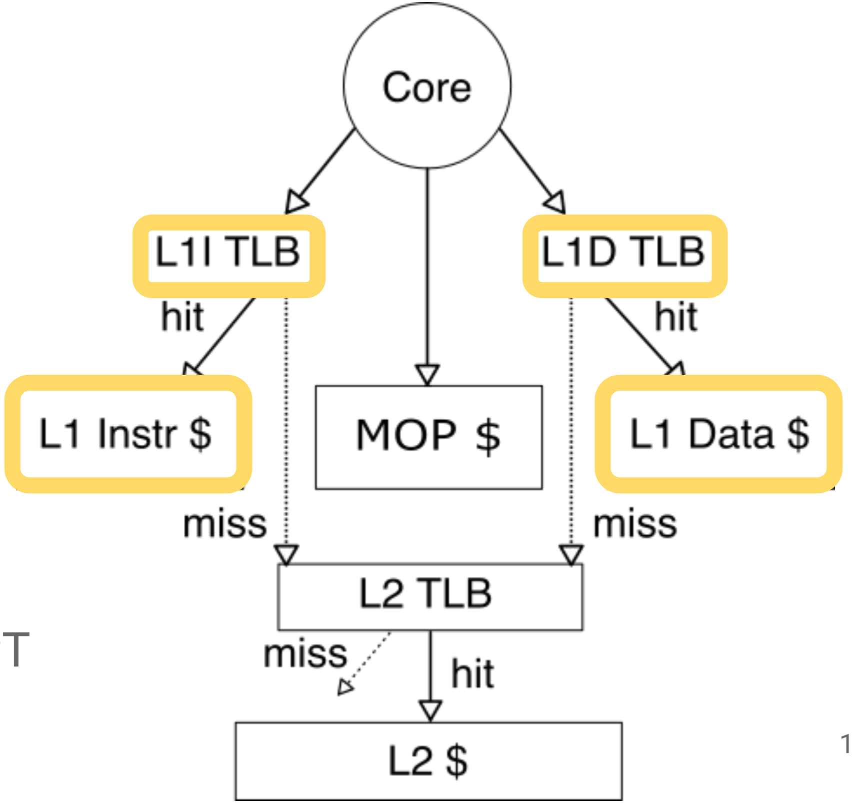
Memory Hierarchy, Cnt’d (2/3)

L1 TLB:

- separate for Data & Instr.
- Fully Associative
- 32 entries
- Access: typically 1 cycle

L1 $:

- separate for Data & Instr.
- 4-way Set Associative
- Cache line: 64 Bytes
- Size: 32/64 KB (configurable)
- Virt. Ind. Phys. Tag. (VIPT) behaves as PIPT
- Pseudo-LRU
- MESI (cache coherence)



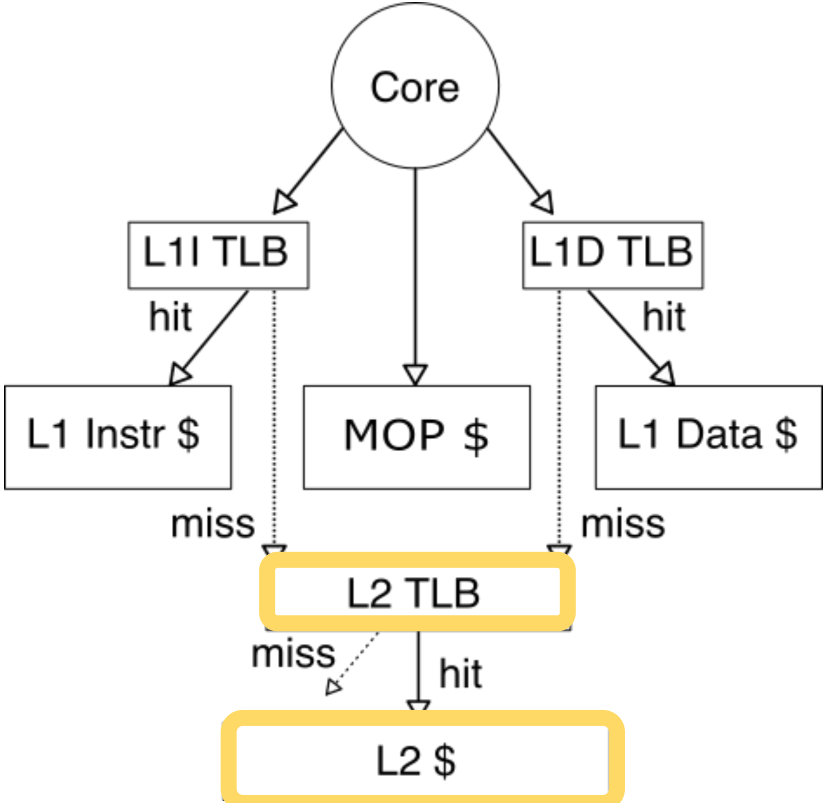
Memory Hierarchy, Cnt'd (3/3)

L2 TLB:

- same for Data & Instr.
- 4-way Set Associative
- 1024 entries
- Access: typically 3 cycles

L2 $:

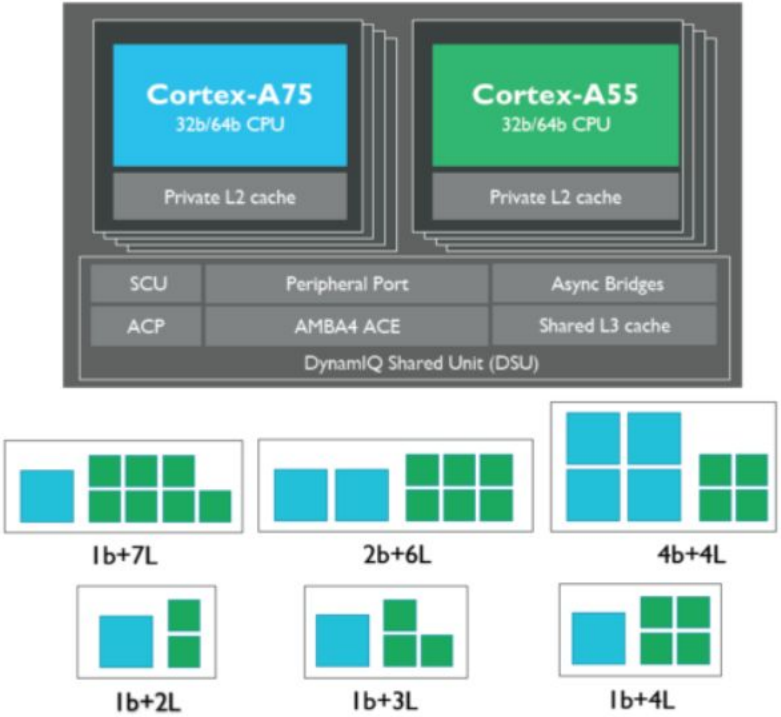
- same for Data & Instr.
- 8-way Set Associative
- Cache line: 64 Bytes
- Size: 256/512 KB (configurable)
- Inclusive (strictly: L1D, weakly: L1I)
- MESI (cache coherence)



big.LITTLE

Heterogeneous processing architecture that uses two types of processor

- ”LITTLE” processors -> maximum power efficiency
  - texting, email, audio
  - Cortex-A53, Cortex-A55
- ”big” processors -> maximum compute performance
  - mobile gaming and browsing
  - Cortex-A73, Cortex-A75, Cortex-A76, Cortex-A77, Cortex-A78



Example DynamIQ big.LITTLE configurations

DynamIQ cluster

Cluster microarchitecture ==>

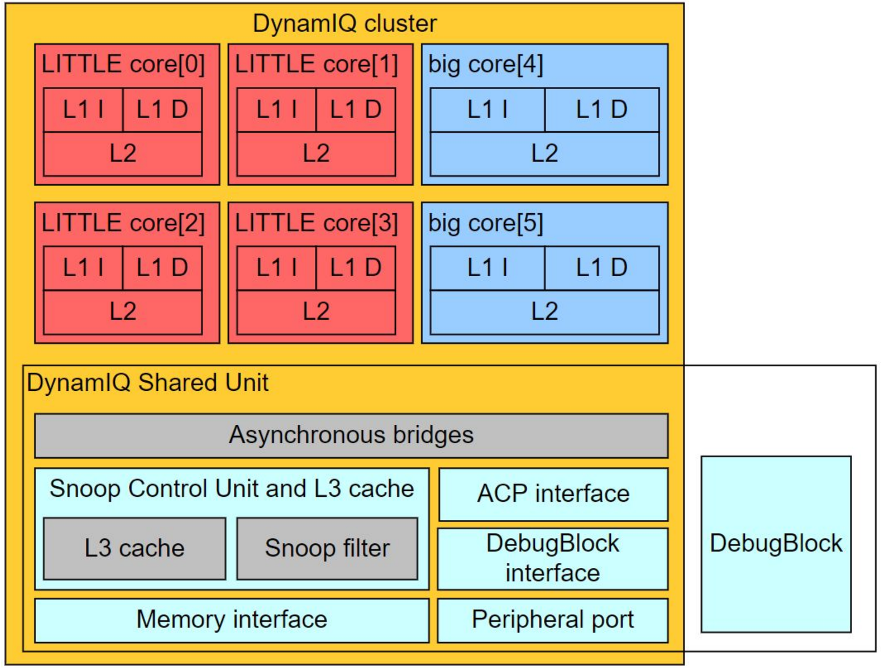
- One or more cores
- DSU

Dynamic Shared Unit (DSU) ==>

- L3 memory system
- Control logic
- External Interfaces

Two configurations ==>

- A set of cores having the same microarchitecture
- Two sets of cores, where each set has a different microarchitecture



DSU Components

L3 cache

- 2MB to 4MB
- 16-way set associative
- 64-byte cache line

SCU (Snoop Control Unit)

- maintains coherency

Power state control

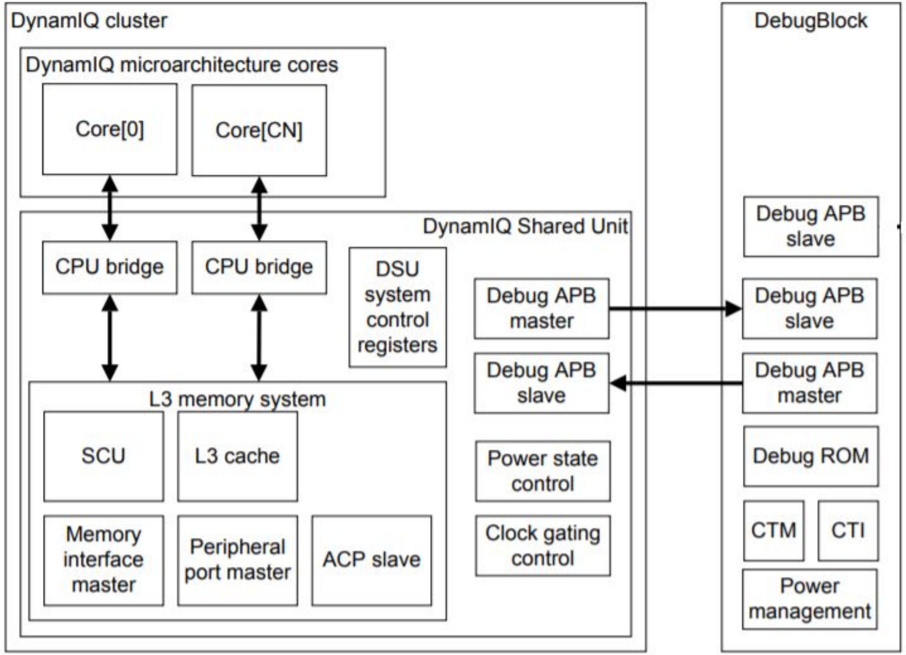
- External power controller

DSU system control registers

- Set of registers accessible from any core in the cluster

ACP slave

- Accelerator Coherency Port

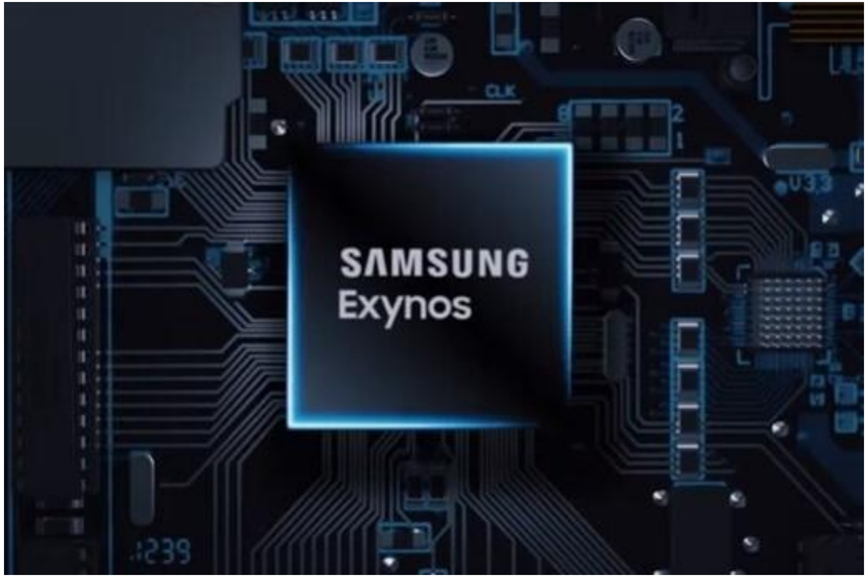


Cortex-A78 with DynamIQ

- Up to eight cores
- Up to four Cortex-A78s may be clustered together
- The cluster may also include up to four additional little cores such as the Cortex-A55 in a big.LITTLE configuration
- One or more of the A78 cores may be swapped out for a Cortex-X1 core in order to achieve even higher performance
- Compared to a quad-core A77 cluster on 7 nm, a quad-core A78 cluster on 5 nm provides +20% sustained performance improvement while reducing the silicon area by about 15%.

Use cases

- AR, XR
- ML, AI
- Edge Computing
- AAA Gaming as an exciting use-case
- A78 CPU + Mali-G78 GPU
    - high-fidelity gaming experiences
    - longer battery life on smartphones for extended and enhanced ‘all-day-play’
    - Samsung Exynos 1080

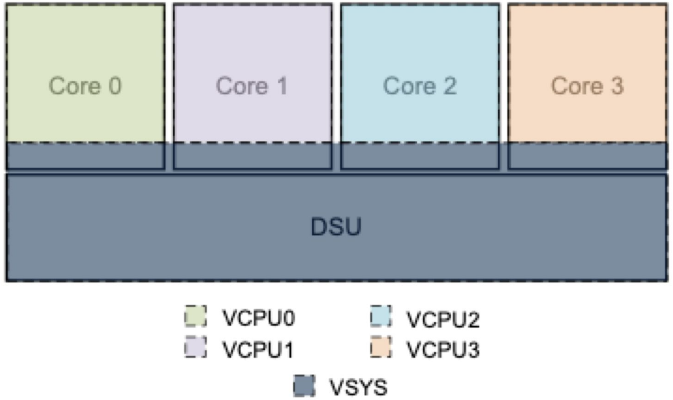


Power Management

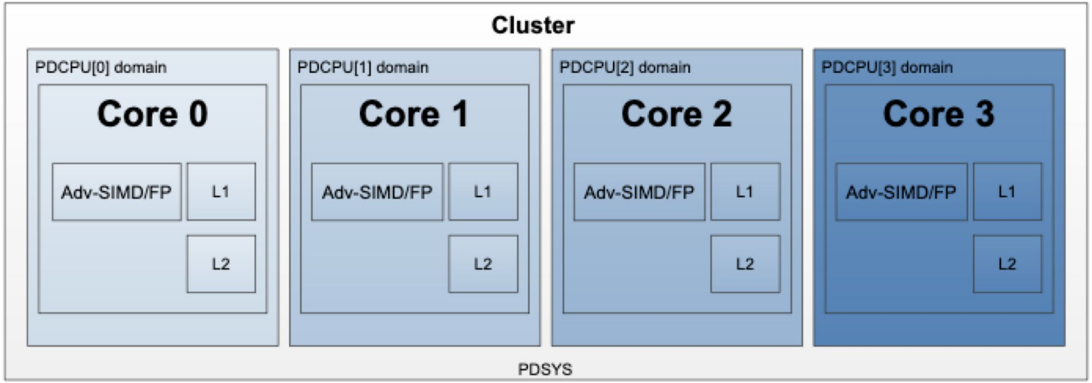
Dynamic: clock gating, DVFS

Static: dynamic retention, powerdown

Voltage Domains



Power Domains



Source of figures: ARM A78 TRM

Energy Efficiency

Various components optimized:

- **Branch prediction**
  - Supports 2 taken branches per cycle
  - Increased accuracy for conditional branches
- **Mid-core and pipeline**
  - More instruction fusions
  - ROB: increased instruction per unit area
- **Memory sub-system**
  - Extra Addr. Generation Unit (AGU): 50% ⇧ load bandwidth
  - Improved data prefetching: memory area coverage, accuracy and timeliness

Source:
1) Arm's New Cortex-A78 and Cortex-X1 Microarchitectures: An Efficiency and Performance Divergence, anandtech.com

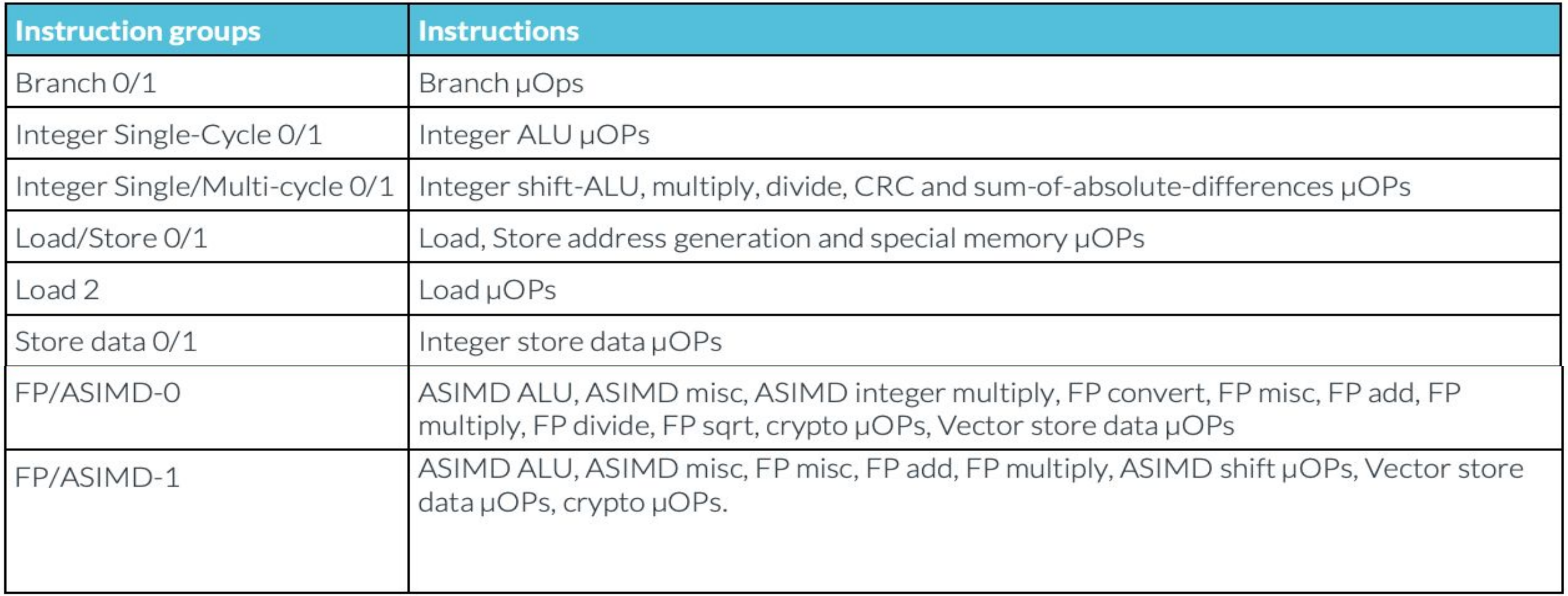
References

1. Arm Cortex-A78 Core Technical Reference Manual, ARM.
2. Cortex-A78 - Microarchitectures - ARM, Wikichip.
3. Arm's New Cortex-A78 and Cortex-X1 Microarchitectures: An Efficiency and Performance Divergence, anandtech.com
4. Arm Unveils the Cortex-A78: When Less Is More, Wikichip.
5. Sustained performance through Arm Cortex-A78 CPU - Processors blog - Processors, ARM.
6. Arm Cortex-A78 Core Software Optimization Guide, ARM.
7. ARM DynamIQ Shared Unit Technical Reference Manual, ARM.
8. Seznec A., “A Case for Two-Way Skewed-Associative Caches”, ISCA 1993.
9. Mutlu O., Comp. Arch., “High Performance Caches”, CMU, Spring 2015.
10. Al-Zoubi H. et al, “Performance evaluation of cache replacement policies for the SPEC CPU2000 benchmark suite”, ACM-SE 2004.
11. Arm's New Cortex-A77 CPU Micro-architecture: Evolving Performance, anandtech.com
12. Cortex-A15 Technical Reference Manual. ARM.

Thank you!

Backup Slides

Cortex-A78 core operations

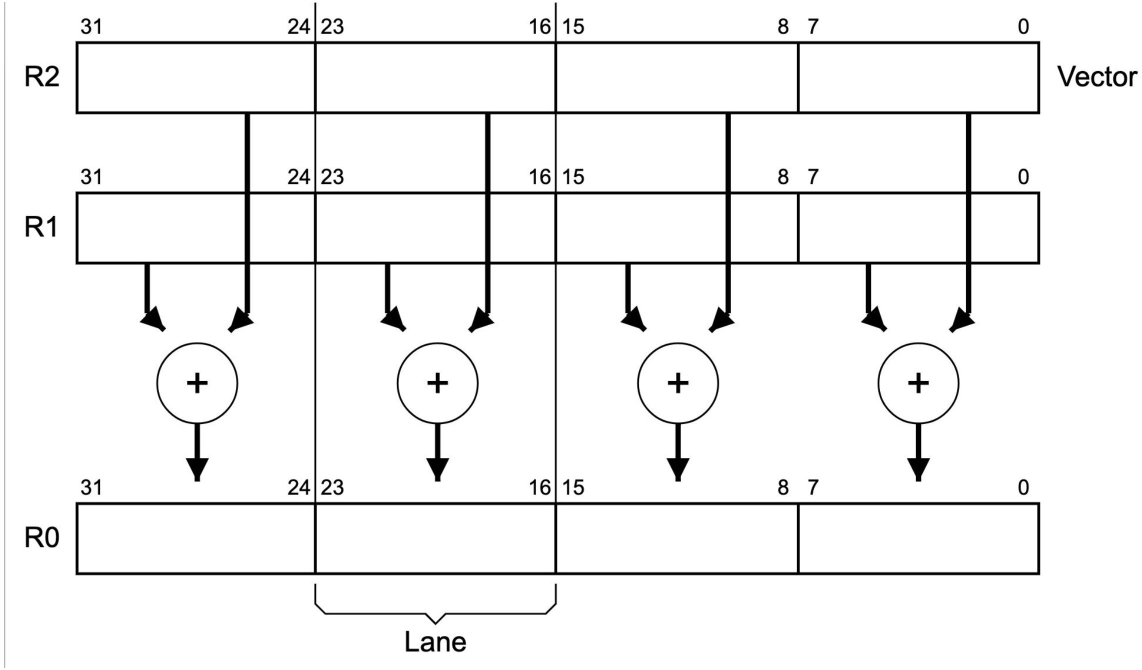


Instruction groups	Instructions
Branch 0/1	Branch µOps
Integer Single-Cycle 0/1	Integer ALU µOPs
Integer Single/Multi-cycle 0/1	Integer shift-ALU, multiply, divide, CRC and sum-of-absolute-differences µOPs
Load/Store 0/1	Load, Store address generation and special memory µOPs
Load 2	Load µOPs
Store data 0/1	Integer store data µOPs
FP/ASIMD-0	ASIMD ALU, ASIMD misc, ASIMD integer multiply, FP convert, FP misc, FP add, FP multiply, FP divide, FP sqrt, crypto µOPs, Vector store data µOPs
FP/ASIMD-1	ASIMD ALU, ASIMD misc, FP misc, FP add, FP multiply, ASIMD shift µOPs, Vector store data µOPs, crypto µOPs.

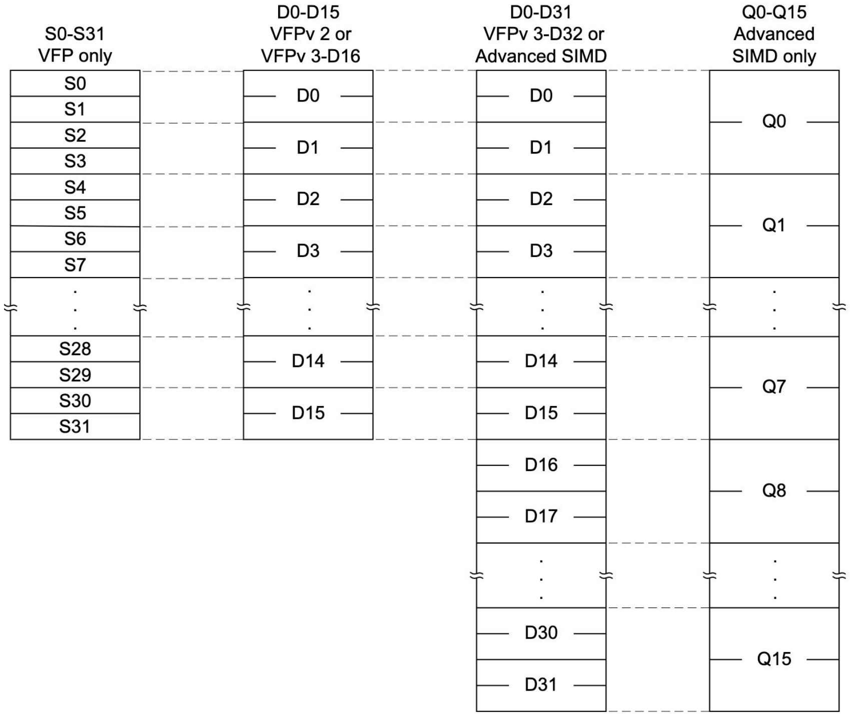
Dispatch Constraints

Execution Unit	# of µOps
Store data (all) / Branch (all)	4
Integer Single/Multi-cycle (all)	4
Integer Single/Multi-cycle 0	2
FP/ASIMD-0	2
FP/ASIMD-1	2
Load (all)	6

SIMD Instructions With NEON

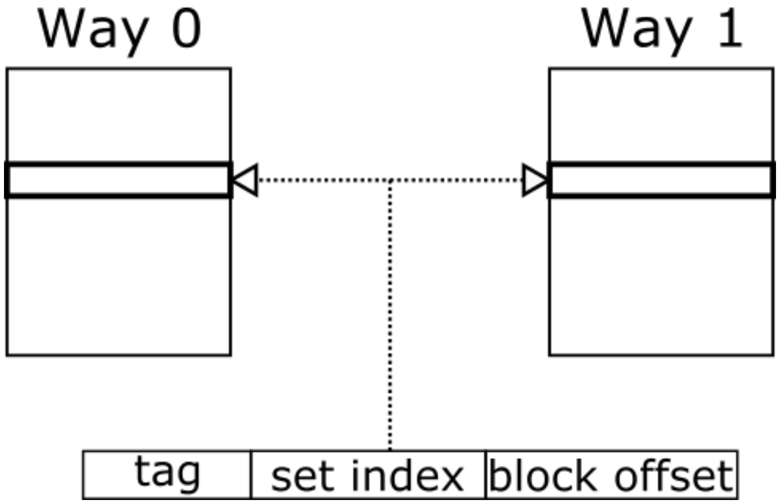


NEON register set

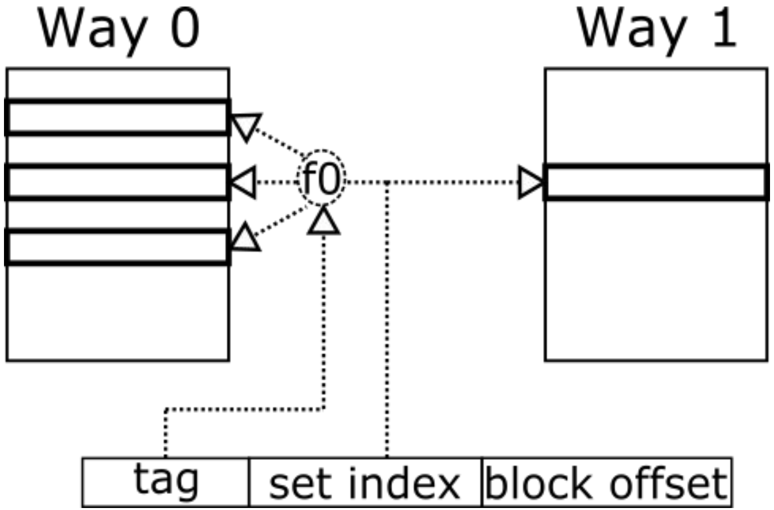


Skewed Associativity

Typical 2-way:



Skewed 2-way:



Key idea: Reduce cache conflicts by randomizing the set index using a different hash function for each way.

Adv.: reduces conflict misses

Disadv.: hash functions induce extra latency

Sources:
1) Seznec A., “A Case for 2-Way Skewed-Assoc. Caches”, ISCA’93
2) Mutlu O., Comp. Arch., High Performance Caches, CMU, Spr’15

Write Streaming Mode

Cache line allocation (linefill): upon a read/write miss.

However, not all writes require allocation: Writes of large blocks of data can unnecessarily **pollute the cache**, e.g. memset(). (waste of energy & performance)

Write Streaming Mode:

- Stores lookup cache, but upon miss they write out to L2 (L3, or DRAM)
- Loads behave as normal

L1 memory system of A78 includes logic to detect when the core has stores pending to full cache line OR upon a DCZVA (full cache line write to zero)

Pseudo-LRU (PLRU)

Key Idea: Replace cache lines based on approximate age rather than the actual age.

Compared to typical LRU, PLRU improves performance.

One simple implementation is called: **Bit-PLRU** or Most Recently Used (MRU)-based

- An MRU bit is assigned to each cache line
- MRU bit is set to 1 upon a hit
- When there is a need for replacement, cache controller looks up to find first cache block with MRU = 0. When found, it is replaced and MRU is set to 1.
- When (almost) all blocks have MRU = 1, last set clears the MRU of the other blocks

Source:
1) Al-Zoubi H. et al, “Performance evaluation of cache replacement policies for the SPEC CPU2000 benchmark suite”, ACM-SE 2004