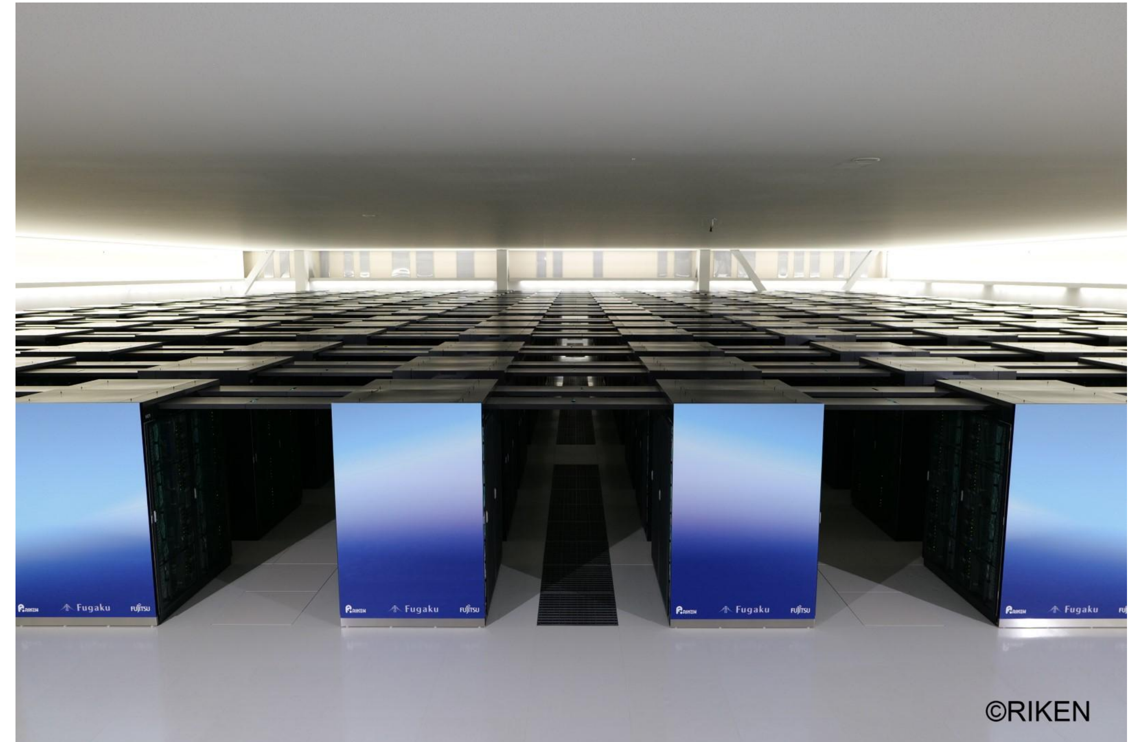
Fujitsu A64FX

Amritesh Dasari,
Jakob Arend,
Pratik Sampat and
Sudhanshu Agarwal

History

- Launched in 2019 for use in the Fugaku supercomputer
- Fugaku was rated as the *most powerful* supercomputer in June 2020 (since been overtaken)

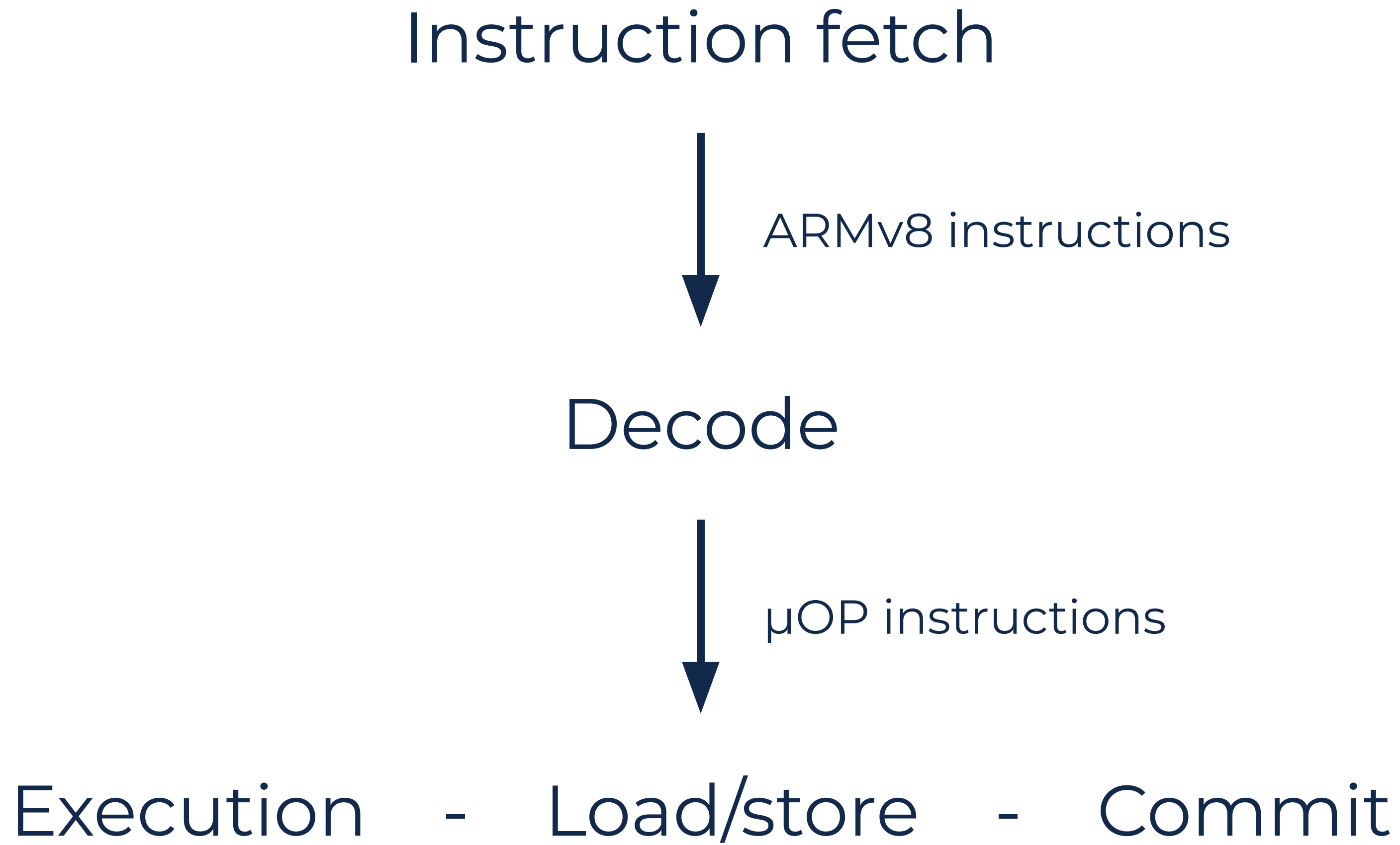


“Supercomputer Fugaku Named World’s Fastest.” Fujitsu Blog - Global, https://corporate-blog.global.fujitsu.com/fgb/2020-06-22/supercomputer-fugaku-named-world-fastest/. Accessed 15 Nov. 2022.

Processor Microarchitecture

Source: A64FX Microarchitecture Manual v1.8

General Pipeline

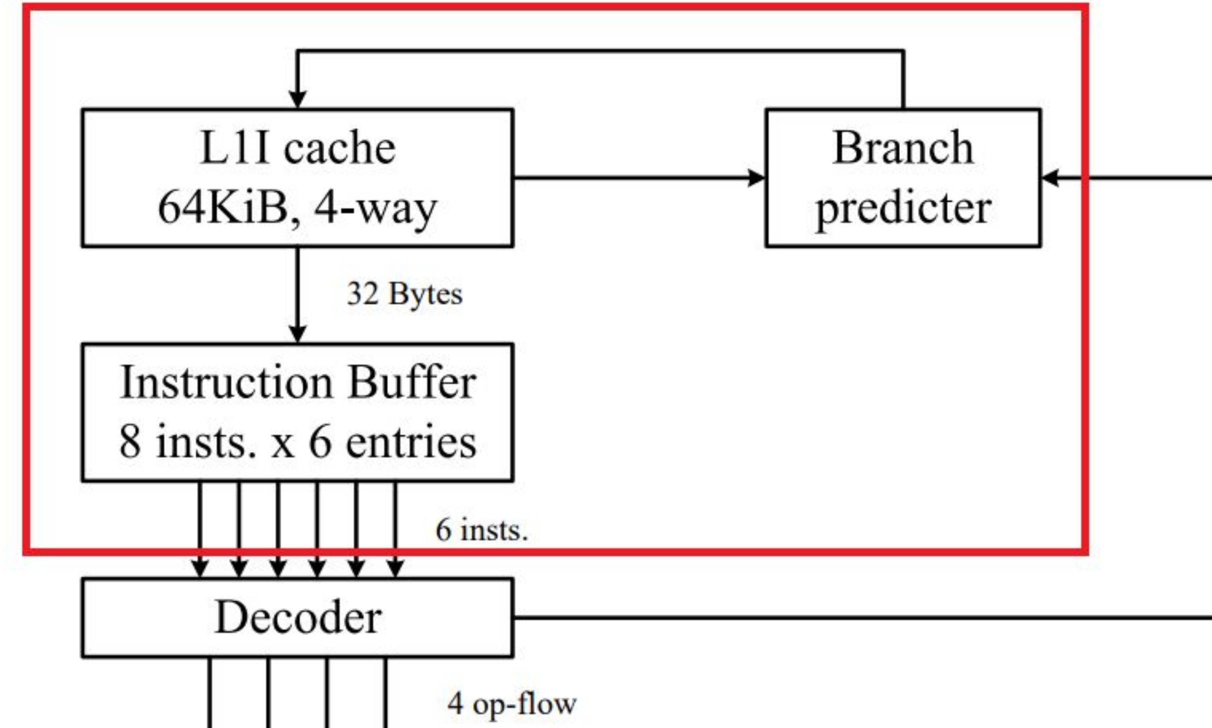


Instruction Fetch Stage

Fetches 8 instructions simultaneously

Predicts four branch directions per cycle

Predicts direction of one taken branch per cycle



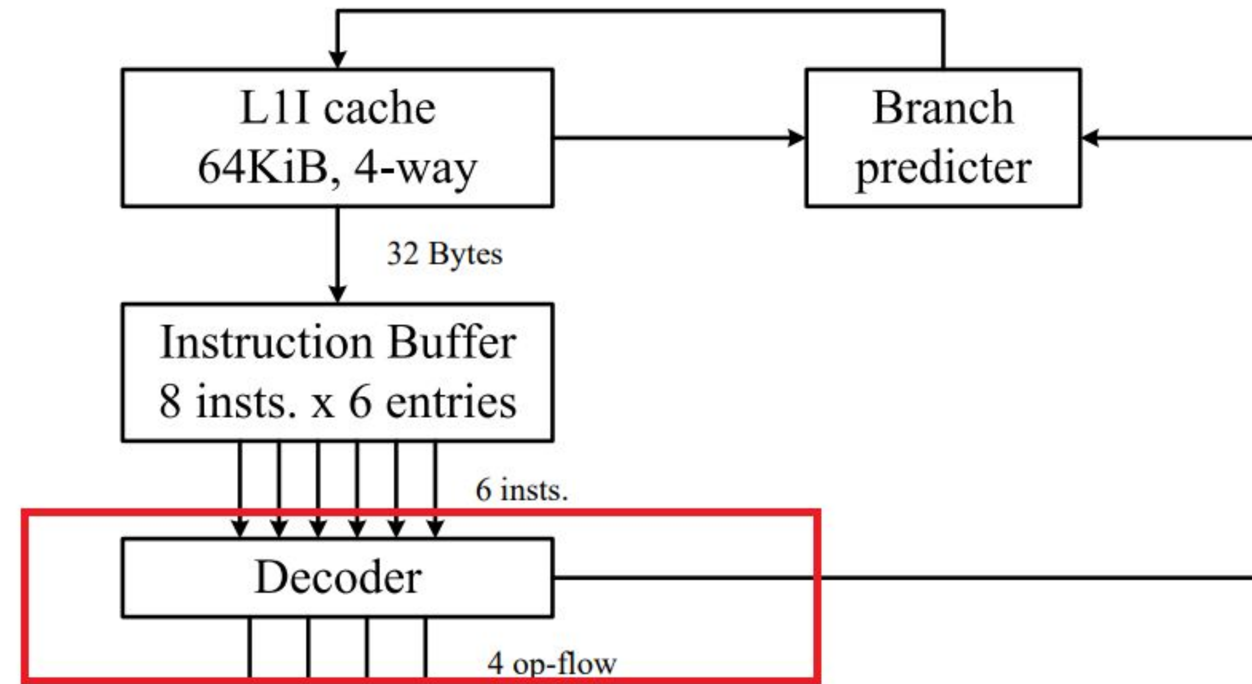
Source: A64FX Microarchitecture Manual v1.8

Decode Stage

Takes in up to 6 ARMv8 instructions per cycle

ARMv8 instructions decoded into µOP instructions

Broken down µOPs sent in-order to execution stage



Source: A64FX Microarchitecture Manual v1.8

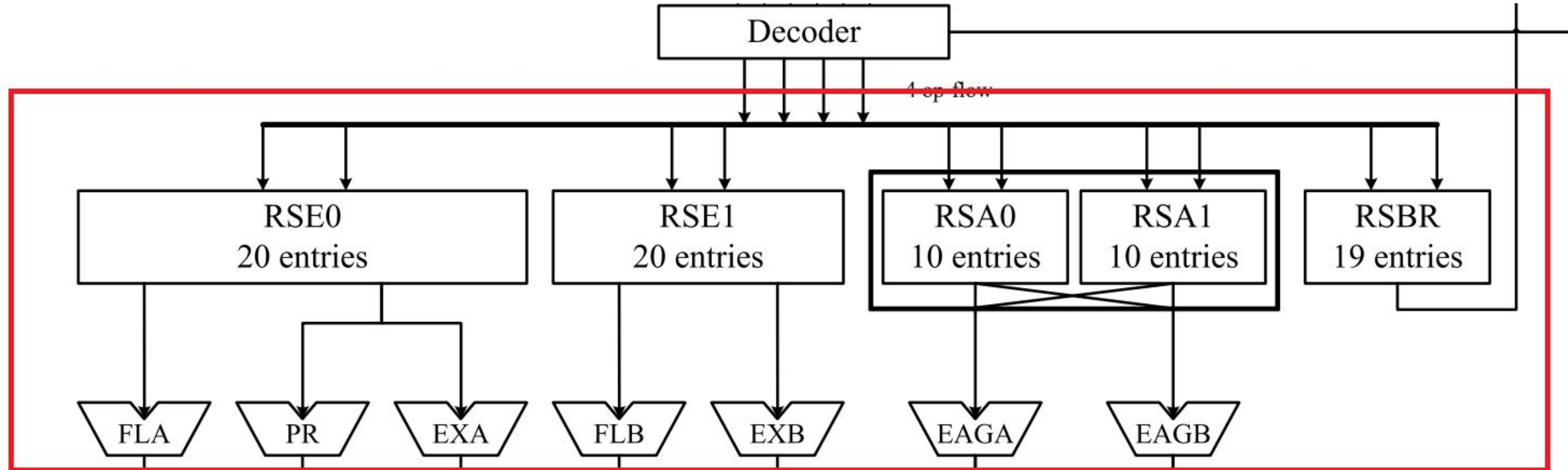
Execution Stage: Overview

Reservation Stations

Execute instructions out of order

Pipelining

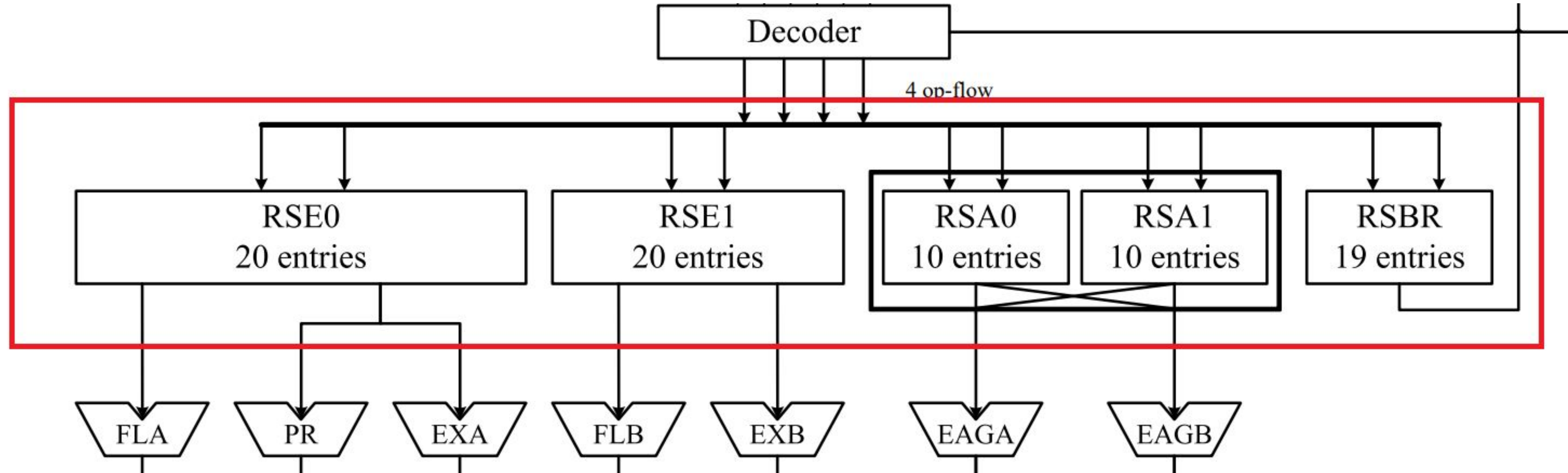
Exploits ILP while executing instructions



Source: A64FX Microarchitecture Manual v1.8

Execution Stage: Reservation Stations

Int/floating/predicate	Address calculation	Branch prediction
2 stations 20 entries each	2 stations 10 entries each	1 station 19 entries

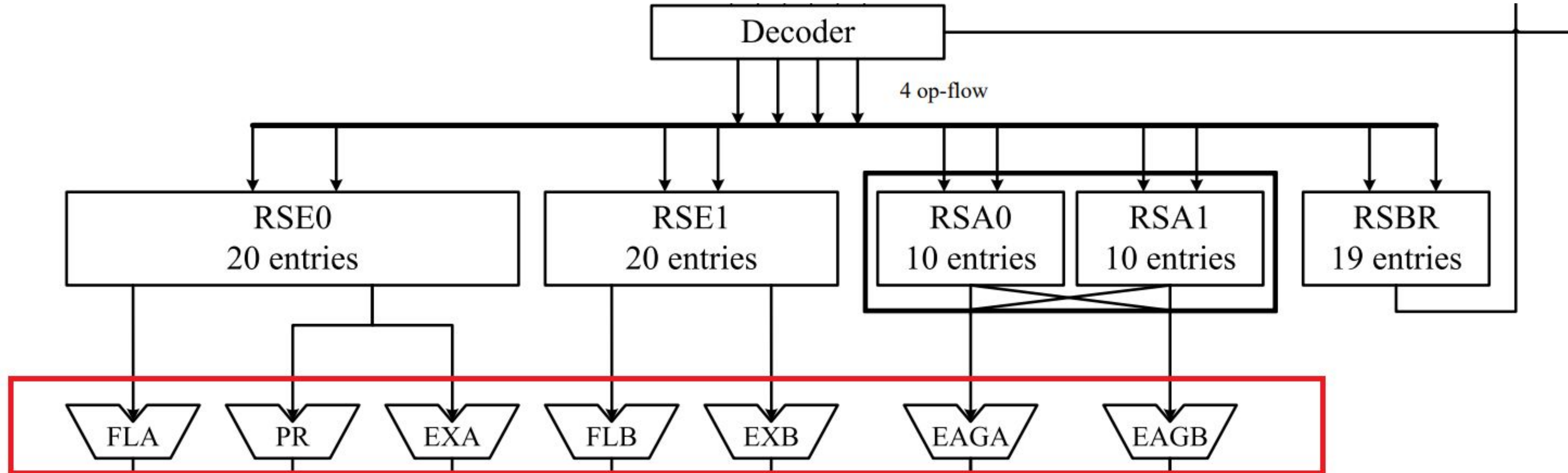


Source: A64FX Microarchitecture Manual v1.8

Execution Stage: Pipelining

Five separate pipelines for integer, SIMD floating point and SVE, predicate, branch, and load/store operations respectively

Pipelines are 11-24 stages deep and support operand bypassing



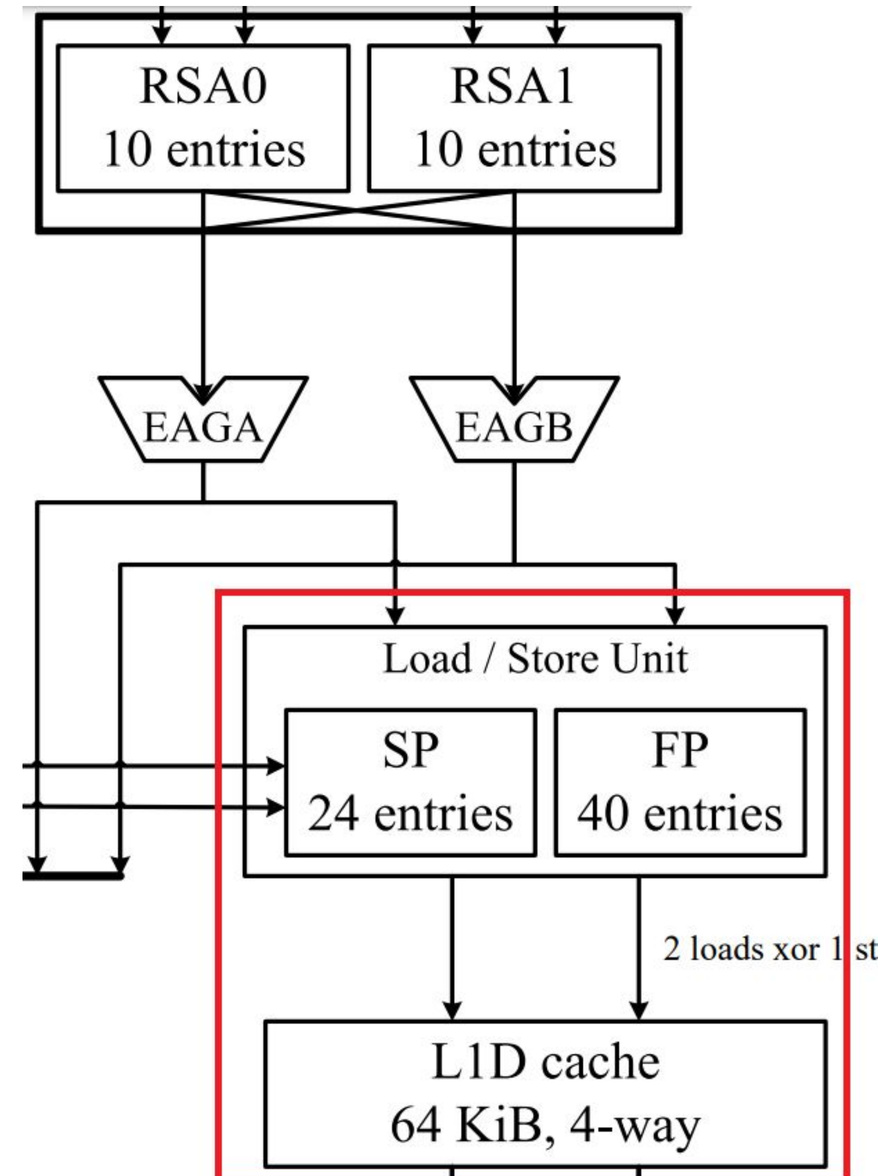
Source: [A64FX Microarchitecture Manual v1.8]

Load/Store Stage

Executes one store or two loads at once

Uses virtual load and store ports

Load/store both pipelined for performance



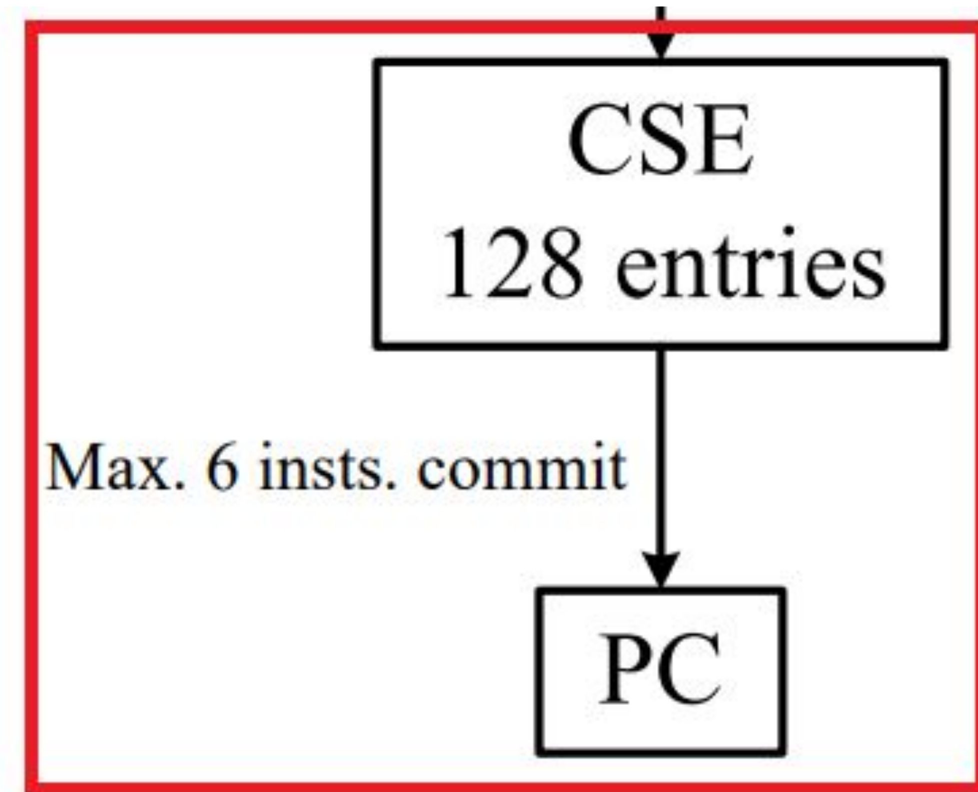
Source: A64FX Microarchitecture Manual v1.8

Commit Stage

Uses a commit stack to maintain proper execution

Checks exceptions and branch prediction results

Commits 4 µOPs per cycle

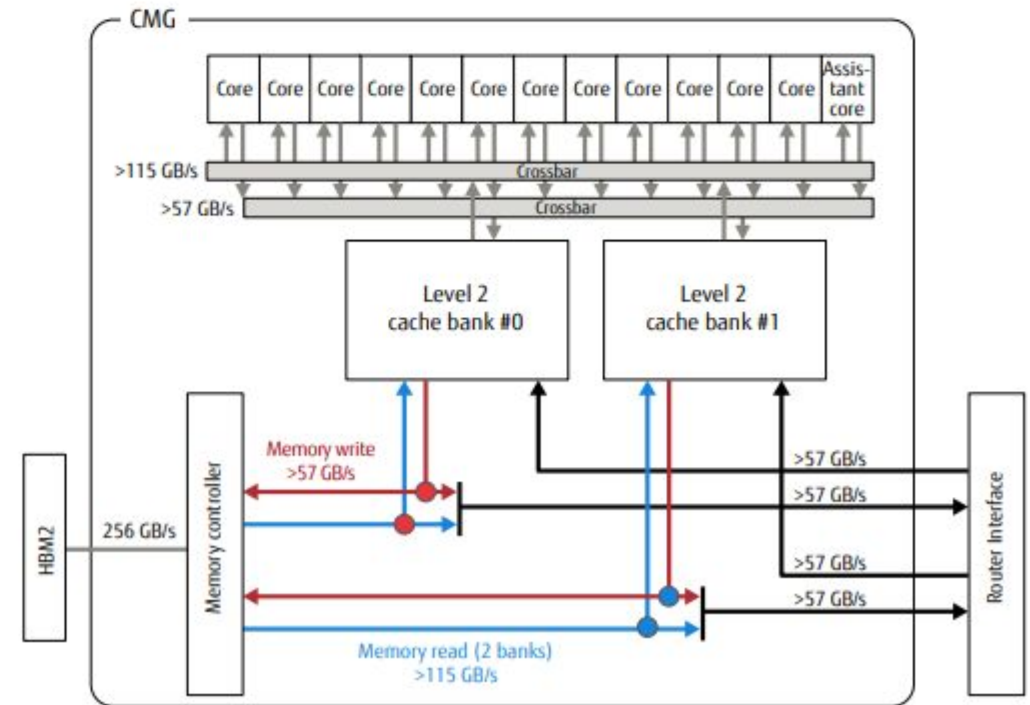


Source: [A64FX Microarchitecture Manual v1.8]

Memory Hierarchy

Memory Hierarchy

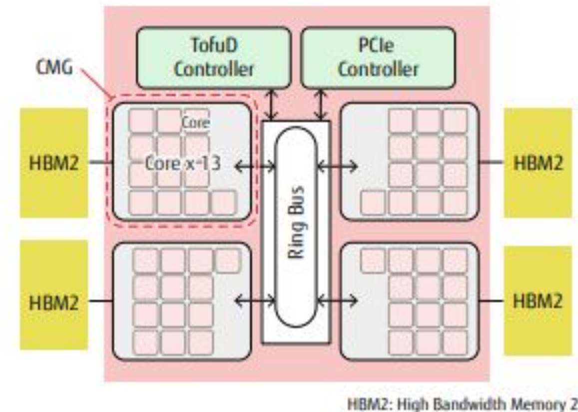
- The processor employs Single Instruction Multiple Data Compute method with 512 bits as the width.
- L1 I-Cache is 3MiB (64KiB/Core)
- L1 D-Cache is 3MiB (64KiB/Core)
- L2 Cache is 32MiB (8MiB*4)
- This 4x multiplier is due to the CMGs that this processor has (*CMG is Core Memory Groups).
- The Memory Accesses between these CMGs is in NUMA config.

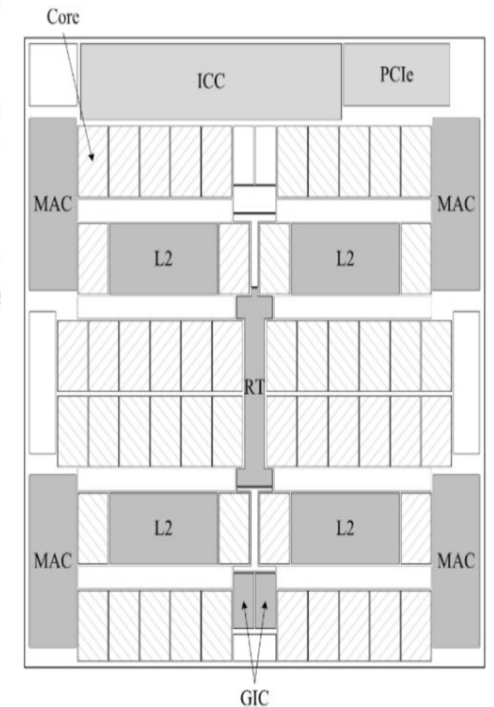


Source: Fujitsu Publication

Scalar Vector Extensions and HMB2 Memory

- This architecture implements 128/256/512 Bits of SVE.
- SVE improves the suitability of the architecture for High-Performance Computing and Machine Learning applications.
- HBM2:
  - Total Capacity: 32GiB
  - No. of stacks per package: 4
  - Memory Capacity: 8GiB
  - Bandwidth: 256GB/s
- One Memory Access Controller per CMG
- Processor known for its Extensive Data Integrity with its Unique 128,400 Error Checkers to detect and correct 1-bit errors on a chip.





Source: Fujitsu Publications

Memory Management Unit

- Translation Lookaside Buffer (TLB)
  - TLB consists of 2 parts: Instruction TLB and Data TLB
  - Each of these have 2 levels:
    - L1: Fully associative and FIFO replacement algorithm.
    - L2: 4-way set associative and LRU replacement algorithm.
- Translation Table Cache
  - A Translation Table Cache is implemented for temporarily storing table descriptors.
  - Diff between TTC and TLB is that TLB is to suppress the occurrence of the table walk and TTC is to reduce latency caused by memory access during table walk.

		For Instruction	For Data
L1	Association method	Full associative	Full associative
	Number of entries	16 entries	16 entries
	Replacement algorithm	FIFO	FIFO
L2	Association method	4-way set associative	4-way set associative
	Number of entries	1,024 entries	1,024 entries
	Replacement algorithm	LRU	LRU

Source: AA64FX Microarchitecture Manual v1.8

Cache Specifications

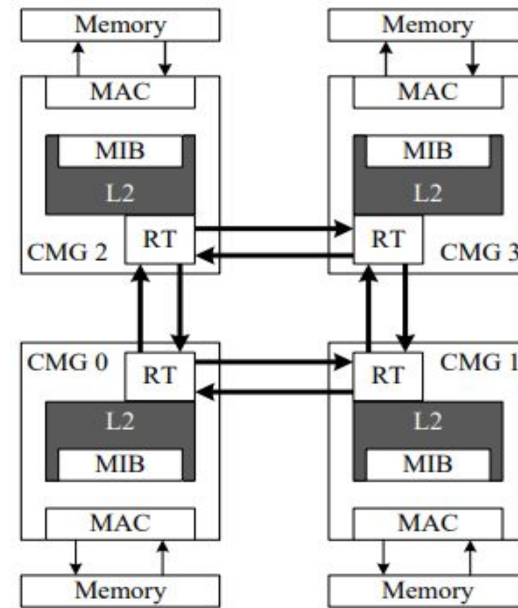
		For Instruction	For Data
L1 cache	Association method	4-way set associative	4-way set associative
	Capacity	64 KiB	64 KiB
	Hit latency (load-to-use)	4 cycles	5 cycles(integer)
			8 cycles (SIMD&FP / SVE in short mode)
			11 cycles (SIMD&FP / SVE in long mode)
	Line size	256 bytes	256 bytes
	Write method	---	Writeback
	Index tag	Virtual index and physical tag (VIPT)	Virtual index and physical tag (VIPT)
	Index formula	index_A = (A mod 16,384) / 256	index_A = (A mod 16,384) / 256
	Protocol	SI state	MESI state

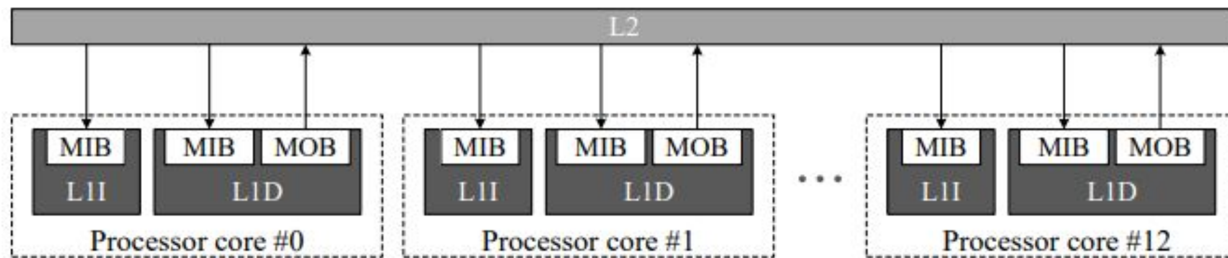
		For instruction and data (by shared)
L2 cache (shared by instruction & data)	Association method	16-way set associative
	Capacity	8 MiB
	Hit latency (load-to-use)	46 to 56 cycles
	Line size	256 bytes
	Write method	Writeback
	Index and tag	Physical index and physical tag (PIPT)
	Index formula	index <10:0> = ((PA<36:34> xor PA<32:30> xor PA<31:29> xor PA<27:25> xor PA<23:21>) << 8) xor PA<18:8>
	Protocol	MESI state

Source: AA64FX Microarchitecture Manual v1.8

Cache Architecture

- L1 caches are implemented in units of processor cores. L2 caches are implemented in units of CMGs
- Cache Coherence explained later.
- A memory unit is connected to only the L2 cache in a CMG.
- Read/Write requests from the L2 cache are sent to the memory unit via the MAC.





Bus Throughput

	Direction	Bus Throughput
L1D	L2 to L1D	64 bytes / cycle (per Core)
	L1D to L2	32 bytes / cycle (per Core)
L2	L2 to L1D	512 bytes / cycle (per CMG)
	L1D to L2	256 bytes / cycle (per CMG)
L2	L2 to L2	64 bytes / cycle (per Ring)
L2	Memory to L2	128 bytes / cycle (per CMG)
	L2 to Memory	64 bytes / cycle (per CMG)

Source: AA64FX Microarchitecture Manual v1.8

Cache Coherence Protocol and Performance

- In the A64FX, coherence between the caches is guaranteed by Hardware. A common MESI protocol is adopted as the protocol for coherence.

Condition		State	Possible Cause of State
M	Modified	Data has been modified from main memory values (Dirty). Other caches at the same level do not have the data.	Data filling due to a store demand request. Stored in a cache line in the E/S state.
E	Exclusive	Data matches main memory values (Clean). Other caches at the same level do not have the data.	Data filling due to a load demand request while other caches do not have the data. Data filling due to prefetch access with a predefined type attribute while other caches do not have the data.
S	Shared	Data matches main memory values (Clean). Other caches at the same level also have the data.	Load demand request in the E state, or data fill request due to prefetch access with the Read attribute.
I	Invalid	A cache line is invalid.	Other caches request data when the data in the E/M state. Data writeback.

The Performance values represent performance per CMG

		Memory Access Performance
Local memory latency (load-to-use)	Shortest core	135.5 ns (@ CPU 2GHz)
	Longest core	144.5 ns (@ CPU 2GHz)
Read throughput	Peak	256 GB/s (per MAC) (@ CPU 2GHz)
Write throughput	Peak	128 GB/s (per MAC) (@ CPU 2GHz)

Source: AA64FX Microarchitecture Manual v1.8

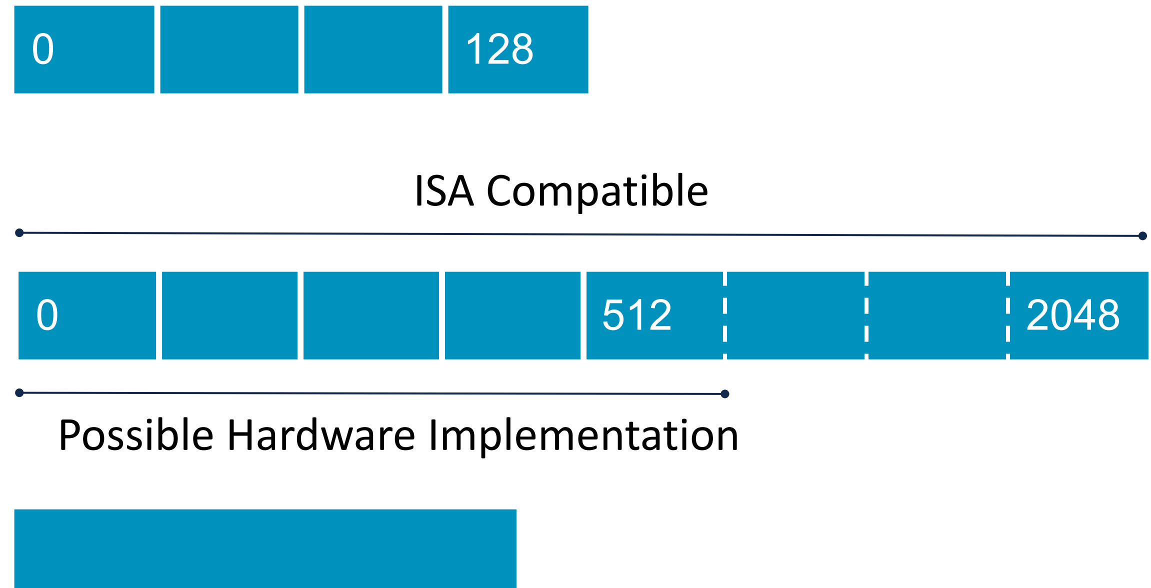
Scalable Vector Extensions

TOFU Interconnect

Scalable Vector Extensions (SVE) for SIMD

- Architectures generally allow only fixed bits for vectorized operations via SIMD
- SVE ISA defines vector length between 128 to 2048 bits and hardware decides the vector length. FX64 defines 512 bits Vector Registers: Z0 - Z31
- Software has no vector length definition. Code can run and scale automatically on different vector length implementations.

  Improves portability

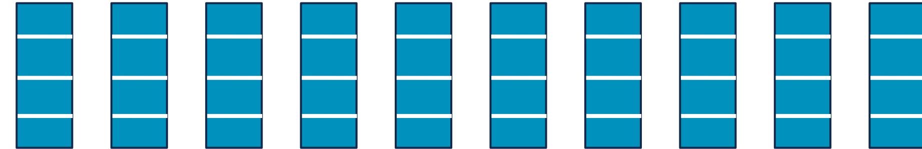


Source: https://ieeexplore.ieee.org/document/7924233

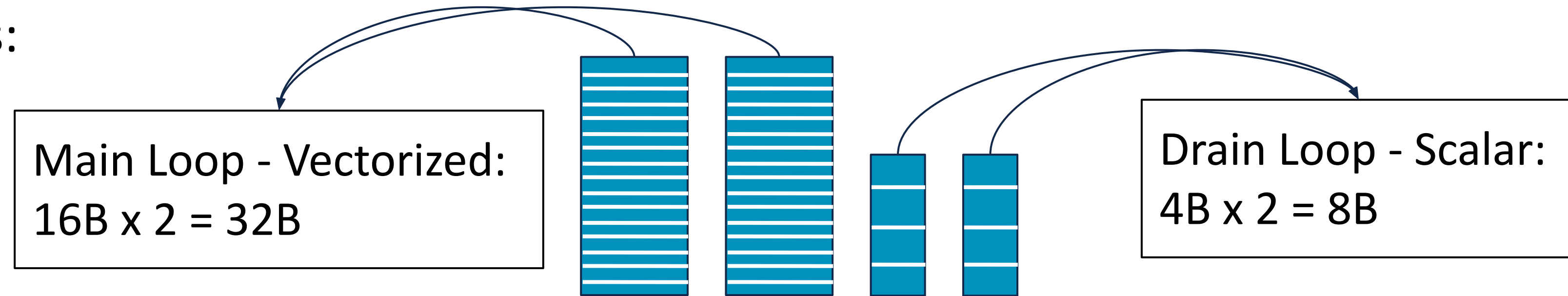
Traditional vectorization techniques vs SVE

> Example: Processing 10 chunks of 4 bytes of data = 40 bytes of data to process

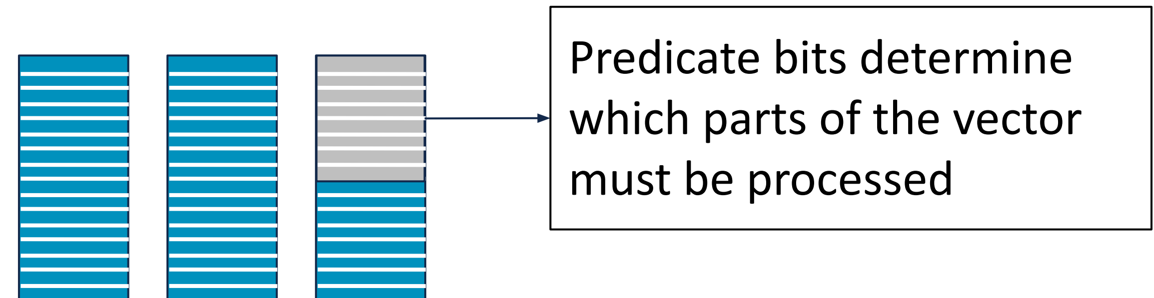
Scalar Architecture
- 10 Iterations over a 4 byte register



Fixed width Vector Architectures:
- 2 iterations Vectorized
  2 iterations Scalar



SVE Vector Length Agnostic
- Adjustable predicate to determine the size of the vector used
- Reduces the binary as there is no need for the scalar loop
- 3 Iterations, all vectorized



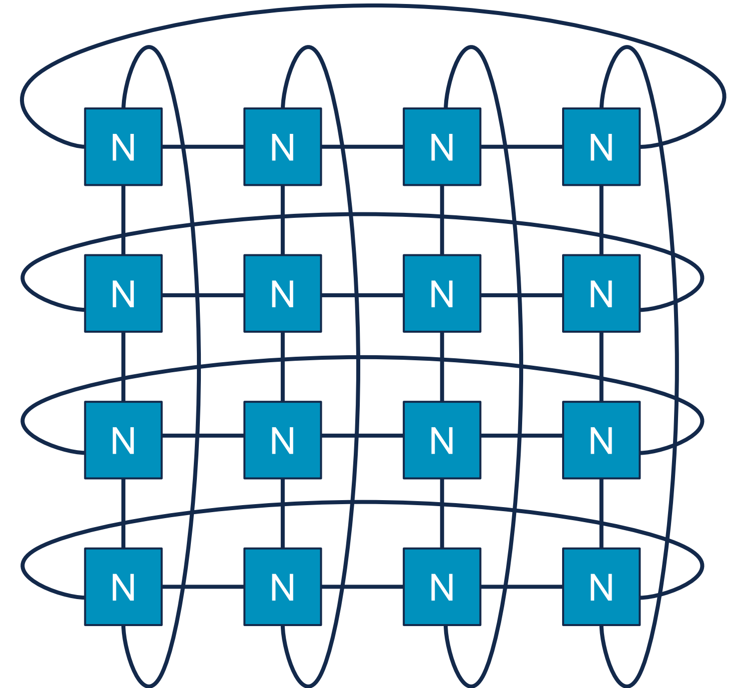
Source: Introduction to Arm SVE

Torus interconnect

- Similar to the Mesh - each node connected to multiple neighbours
- Performance of mesh may depend of placement in the middle or edge
- The Torus connects the edges as well

\+ Has higher path diversity
  \+ Leads in lower hops
  \+ Better Fairness

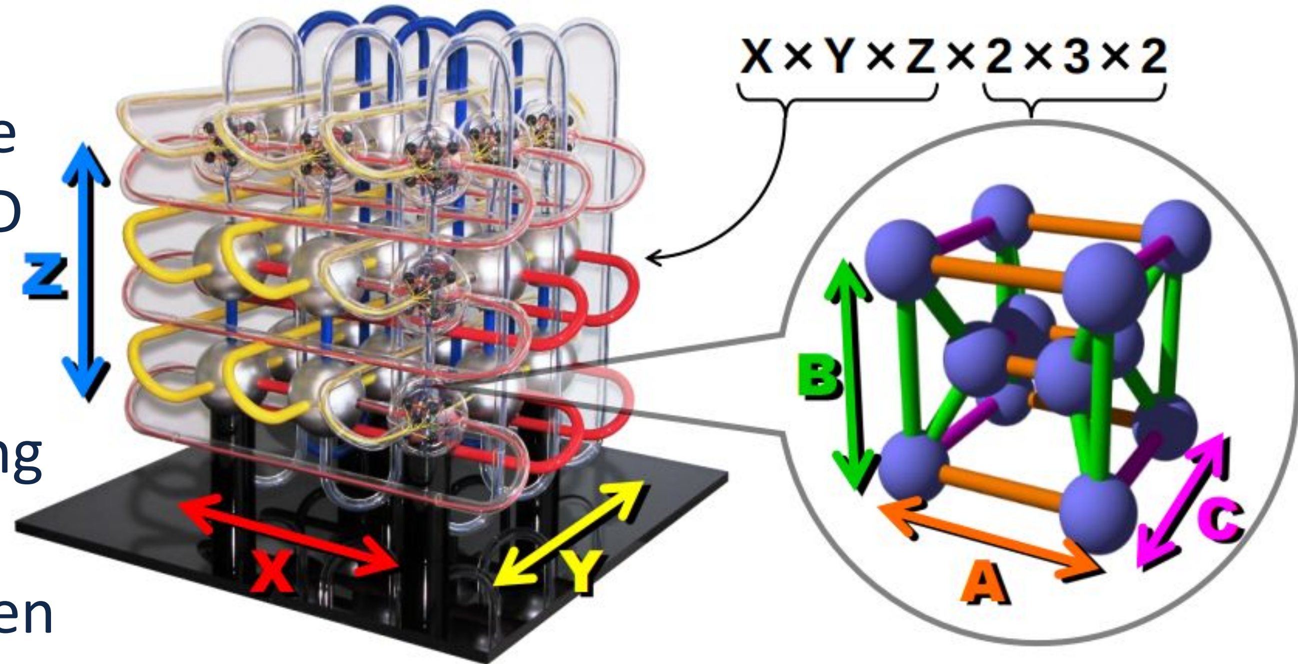
\- Complex - Difficult to have equal link lengths
\- Cost



Source: [Interconnects Lecture - ETH Zurich]

TOFU (Torus Fusion) Interconnect

- 6D Network to appear as 3D Torus
  - Groups of ABC 3D torus with size 2 × 3 × 2 connected by an XYZ 3D torus
- Topology aware: Torus Rank Mapping
  - Give Rank to processes to optimize communication between nearest neighbours
- Highly fault tolerant design



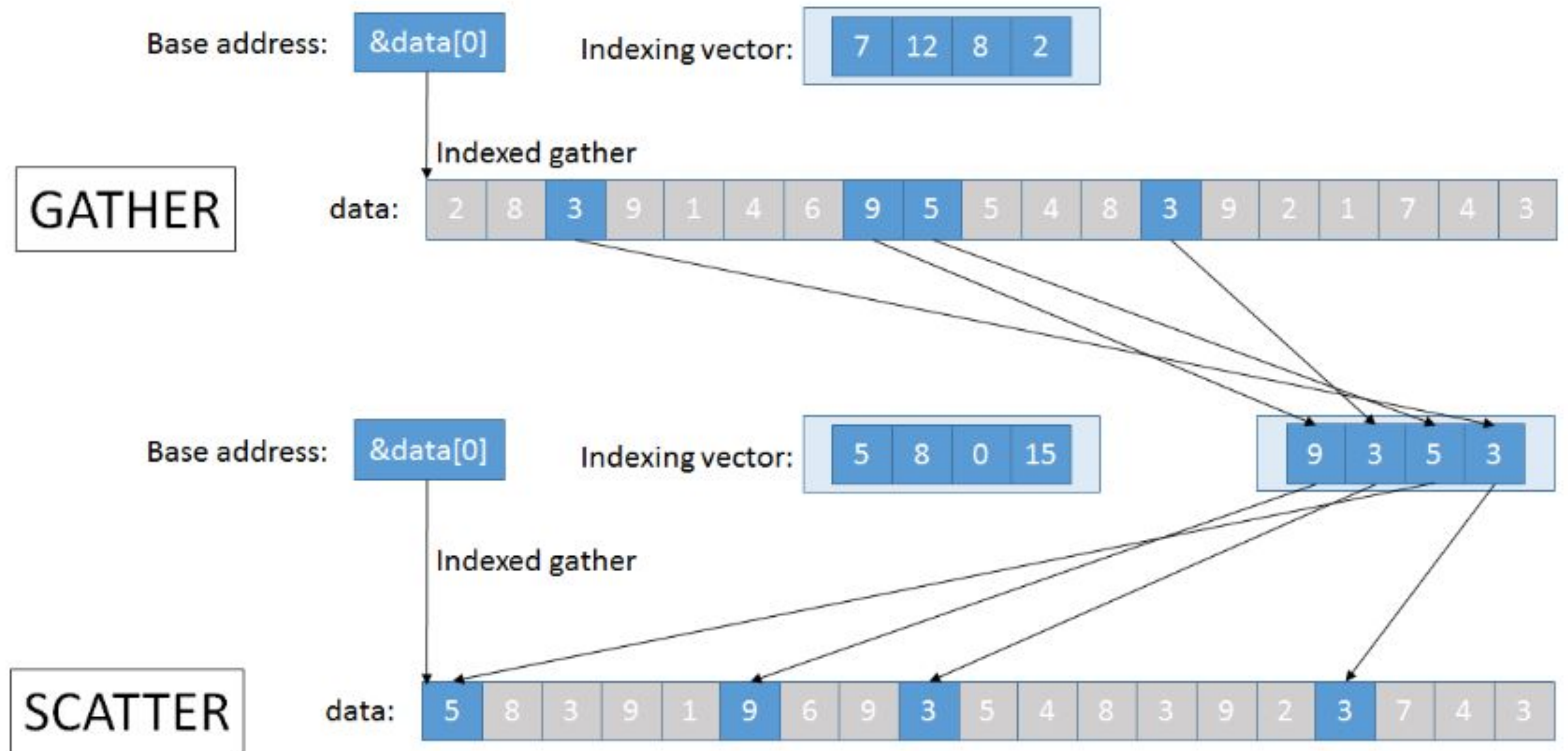
- Can scale over 10,000 nodes
- Interfaced using Open MPI

Source: https://ieeexplore.ieee.org/document/6041538

Combined Gather

Gather and Scatter

Non-sequential data accesses commonly used in sparse vector linear algebra operations

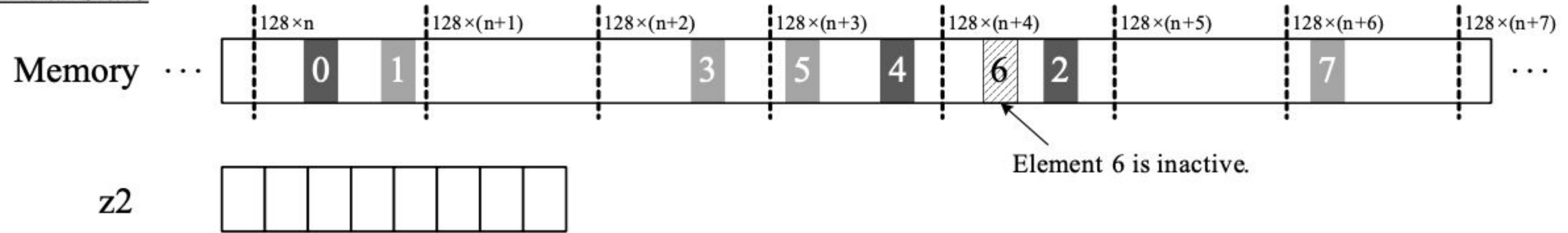


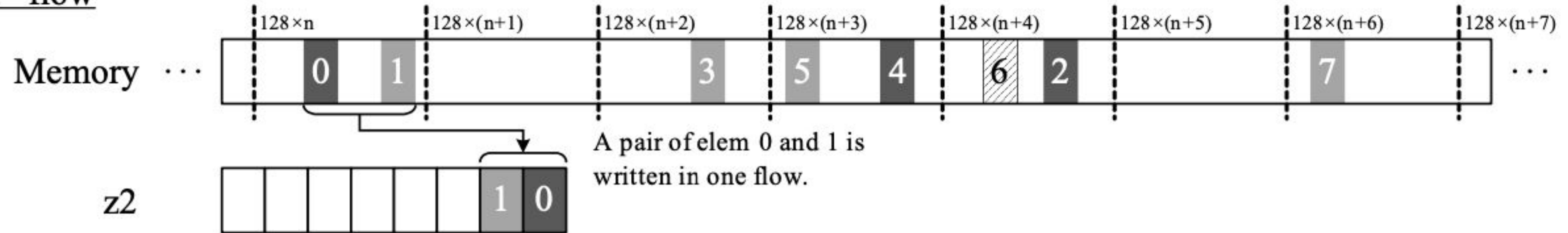
Moving Data - Algorithmica. https://en.algorithmica.org/hpc/simd/moving/. Accessed 18 Nov. 2022.

Combined Gather

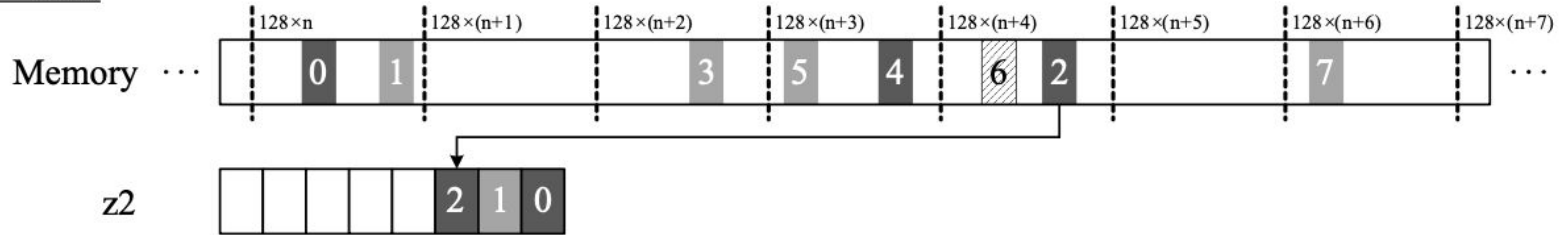
- A64FX optimizes for gather operations by dividing vector elements into 128-bit spaces and loading pairs of data from these spaces concurrently
- Essentially, if two elements lie in the same 128-bit space, they are loaded together
- So, in an ideal case, the latency for gather accesses is halved

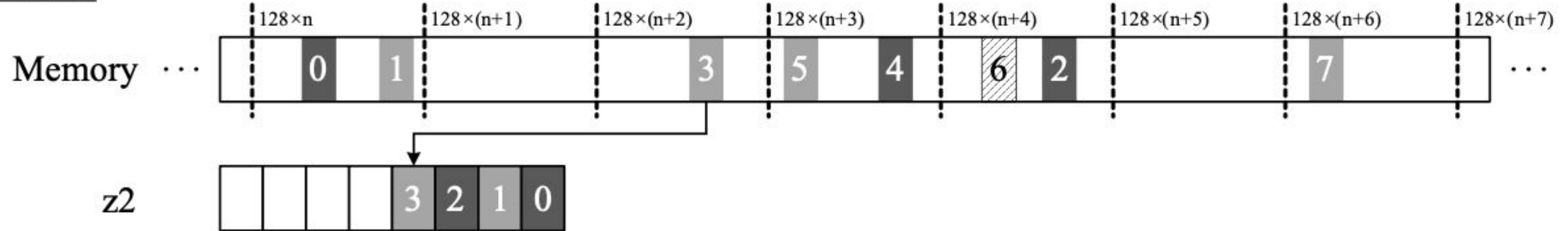
```
ld1d z2.d, p3/z, [x0, z1.d]
```





A64FX Microarchitecture Manual, 2022.





A64FX Microarchitecture Manual, 2022.

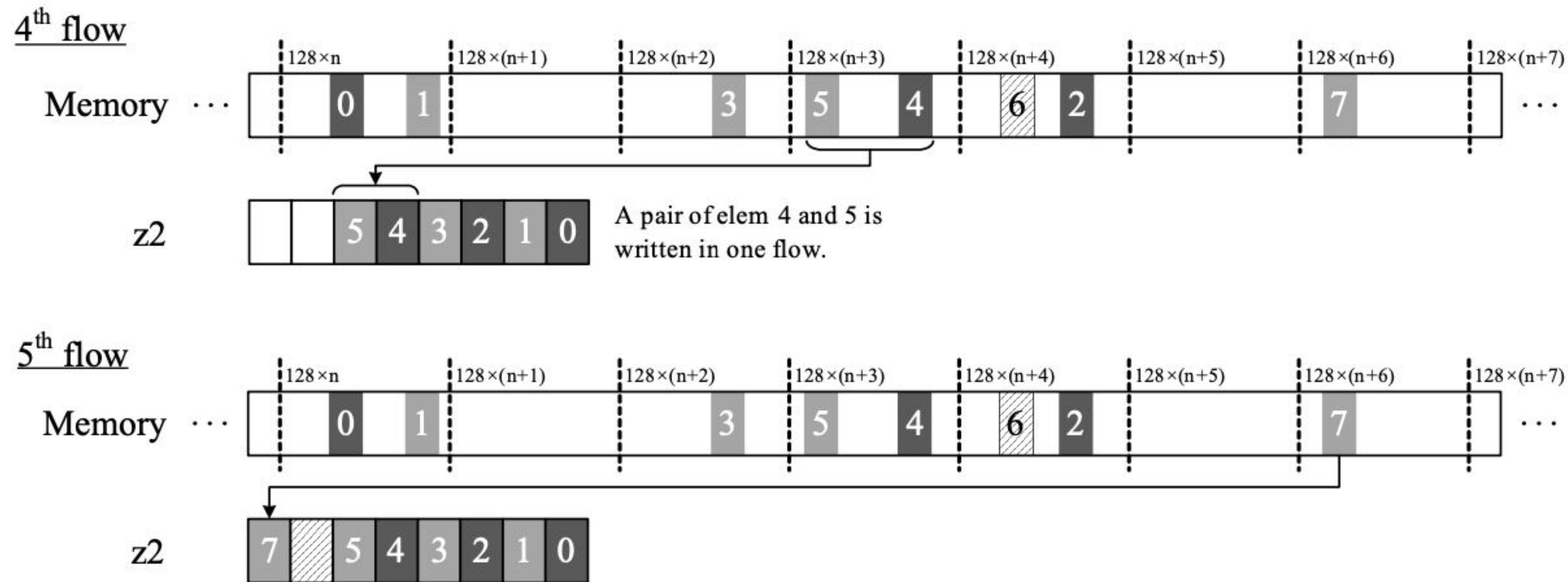


Figure 7-9 Summary of Elements for Gather Instruction

A64FX Microarchitecture Manual, 2022.

Single Core Performance and Power Efficiency

Focus on single core performance

Table 3-1 Branch Predictors of Branch Prediction Mechanism

Role	Branch Predictor
Branch direction & Branch target address prediction	Small Taken Chain Predictor (S-TCP)
Branch direction prediction	Branch Weight Table (BWT) Loop Prediction Table (LPT) Return Address Stack (RAS)
Branch target address prediction	Branch Target Buffer (BTB)

A64FX Microarchitecture Manual, 2022.

- A64FX focuses on improving single core performance by using complicated branch predictors

Small taken chain predictor

- Detects a chain of *taken* branches and prefetches instructions. This is useful when executing long running loops.

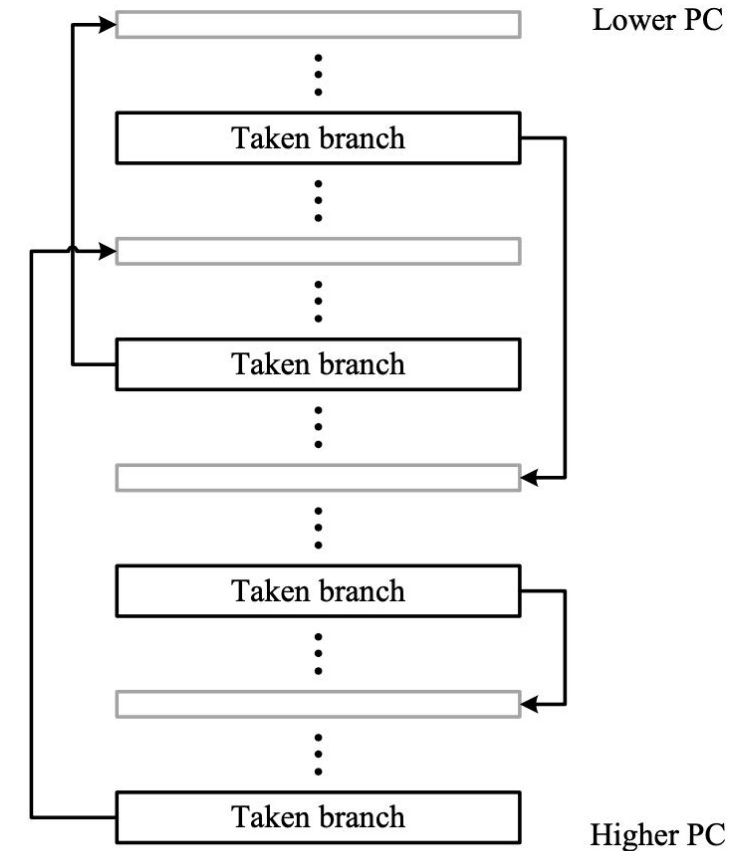
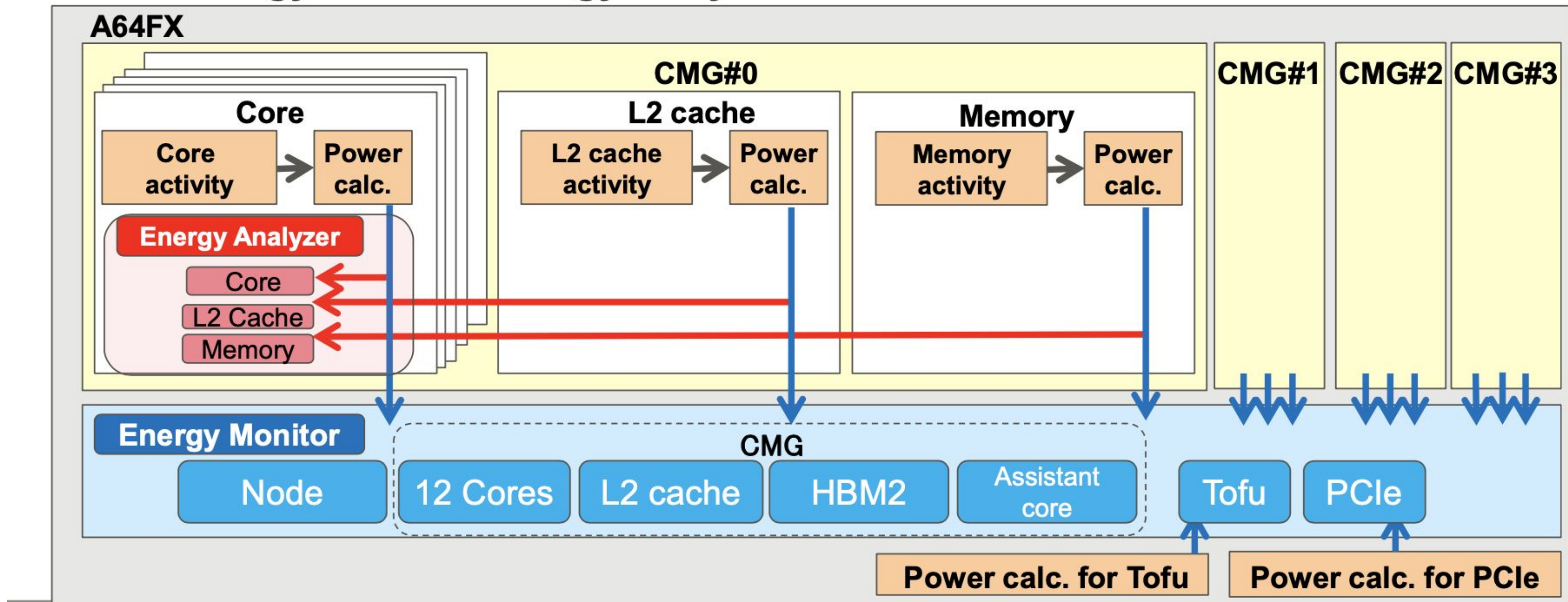


Figure 3-3 Chain Structure Consisting of Multiple Taken Branch Instructions

A64FX Microarchitecture Manual, 2022.

Energy Efficiency

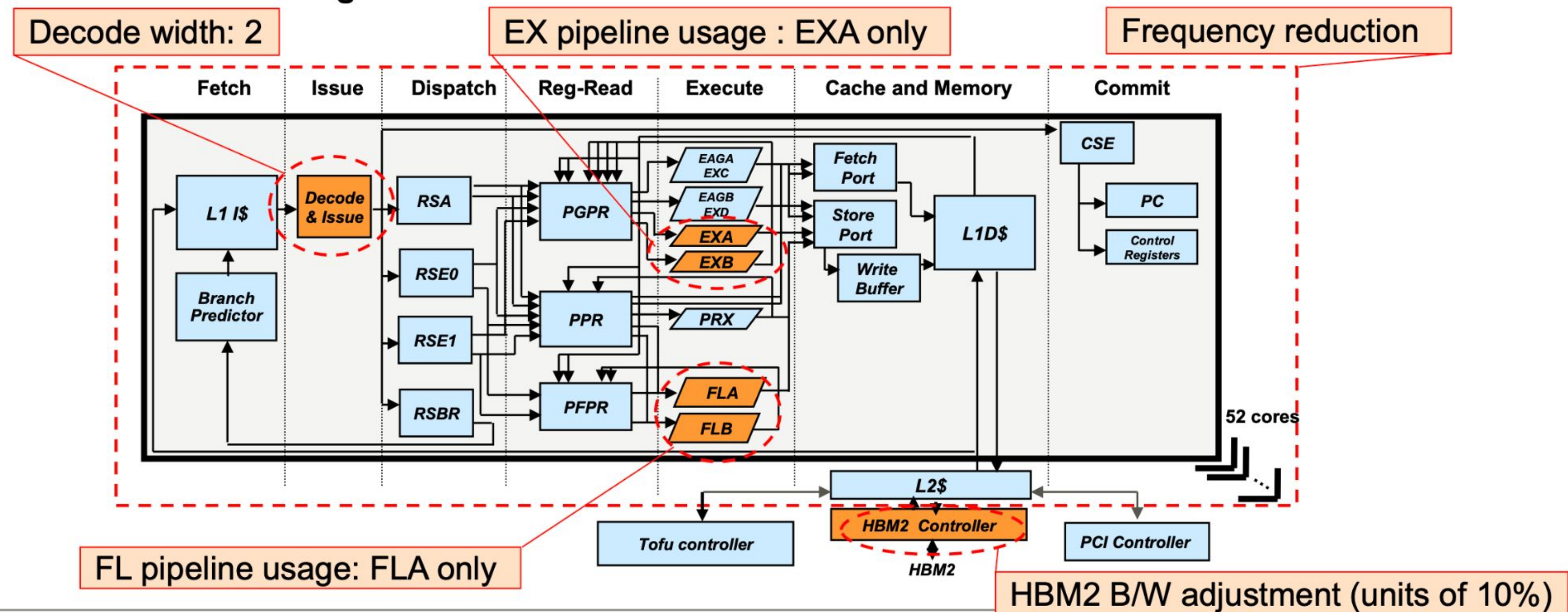
- Fine grained energy analysis for CPU cores and caches



Fujitsu High Performance CPU for the Post-K Computer. Toshio Yoshida, 2018

Power Tuning

- Application level hardware control for power optimization using built-in counters



Fujitsu High Performance CPU for the Post-K Computer. Toshio Yoshida, 2018

Any questions?

References

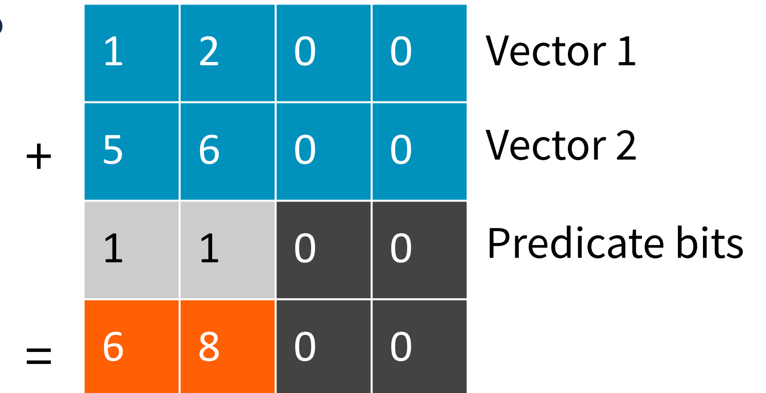
- Memory Hierarchy: https://www.fujitsu.com/global/documents/about/resources/publications/technicalreview/2020-03/article03.pdf
- The ARM Scalable vector Extensions: https://ieeexplore.ieee.org/document/7924233
- Introduction to ARM SVE: https://www.youtube.com/watch?v=eGCcPo4UAHs
- Optimizing HPC Applications using SVE: https://developer.arm.com/documentation/101726/0400
- The Tofu Interconnect: https://ieeexplore.ieee.org/document/6041538
- https://www.fujitsu.com/global/imagesgig5/the-tofu-interconnect-d-for-supercomputer-fugaku.pdf
- A64FX Microarchitecture Manual v1.8, 2022: https://github.com/fujitsu/A64FX/blob/master/doc/A64FX_Microarchitecture_Manual_en_1.8.pdf
- Fujitsu High Performance CPU for the Post-K Computer. Toshio Yoshida, 2018
- Moving Data - Algorithmica. https://en.algorithmica.org/hpc/simd/moving/, Nov. 2022

Backup slides

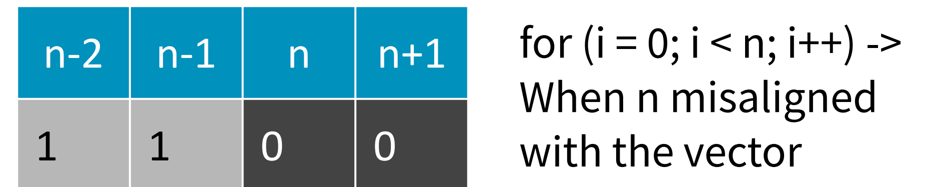
Improvements to auto vectorization

> How to program when we do not know how wide the vectors are?

- Per Lane prediction
  - Map operations to the lane of the vector



- Predicate driven loop control
  - Process partial vectors so that loops heads and tails are not performed in scalar



- Fault-tolerant speculative vectorization:
  - Suppress memory faults if it is not invoked by the first element of the vector

Source: Introduction to Arm SVE