Introduction to the Fujitsu A64FX

The 3rd Isambard hackathon - "Full Steam Ahead!"

Fabrice Dupros Fabrice.Dupros@arm.com

Conrad Hillairet Conrad.Hillairet@arm.com

Phil Ridley Phil.Ridley@arm.com

23rd March 2021

Agenda

- Overview of Arm
- Current Arm-based HPC
- Introduction to the A64FX
- Execution
  - FMLA example
- Cache
  - Stream example

Arm's HPC Field Engineering Team (HPC-FE)

Porting, Tuning, Training, and Enablement

- Application performance engineering
  - Get help optimizing for maximum performance
- System tuning
  - Tune HPC system parameters for your workloads
- Hackathons and tutorials
  - Education, mentoring, and hands-on events to help jumpstart HPC developers

70%

of the world's population uses Arm technology

CPU Engagement Models with Arm

Arm IP is the basic building block for extraordinary solutions.

Core License

- Partner licenses complete microarchitecture
- CPU differentiation via:
  - Configuration options
  - Wide implementation envelope with different process technologies

Arm IP

Architecture License

- Partner designs complete microarchitecture
- Clean room, scratch
- Maximum design freedom:
  - Directly address needs of the target market
- Arm architecture validation preserves software compatibility

Pizza Engagement Models

The basic building blocks for an extraordinary pizza!

Core License

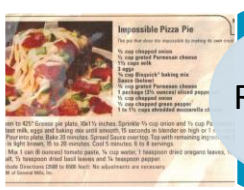
Arm design

Partner IP

Yum!

IP = Irresistible Pizza

Architecture License



Recipe

Ingredients

Yum!

arm

Current Arm-based HPC

Current Deployments

Isambard
Isambard 2
Bristol (UK)

France

Rhea based
Arm Neoverse V1
Europe

Germany

Fugaku
Riken (Japan)

Spain

Armida
Italy

The Cloud

Open access to server class Arm

HPC wire

November 28, 2018

AMPERE.

arm

Introduction to the A64FX

Fujitsu A64FX: Fastest Supercomputer on the Nov'20 Top500

Academia

- Nano-science
- Particle physics

Government

- Long-range forecasting
- Disaster prevention

Oil and Gas

- Exploration and production
- Seismic analysis

Manufacturing

- Structural analysis
- Aerodynamics
- Computational fluid dynamics
- Crash test simulations

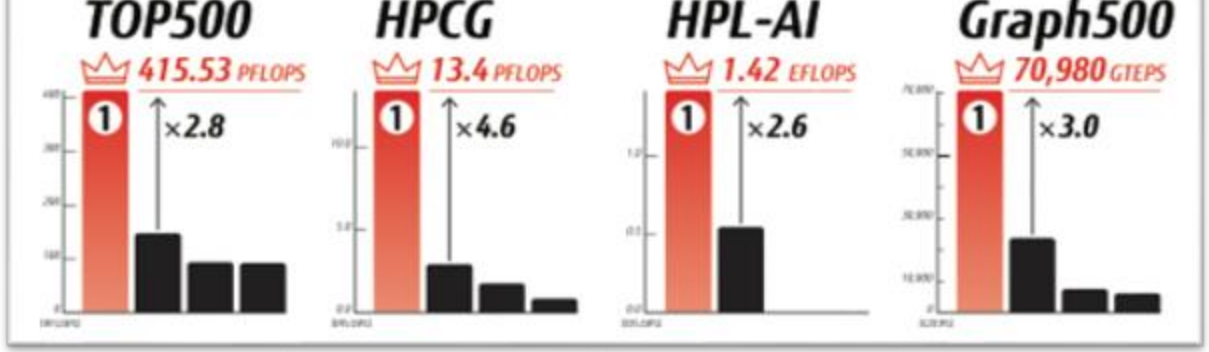
Systems Powered by A64FX

Supercomputer **Fugaku**

FUJITSU Supercomputer
PRIMEHPC FX1000
PRIMEHPC FX700

Awards

Fugaku Achieved First Place in Four Major Supercommputer Rankings (June 2020)



arm

A Leadership CPU from start to finish

Expect excellent performance; expect to have to work to get it

Commodity HPC

- Mainline design
- Common assumptions hold
- Significant fraction of peak without tuning

Leadership HPC

- Codesigned for specific application
- Common assumptions may hurt performance
- Significant tuning effort may be required

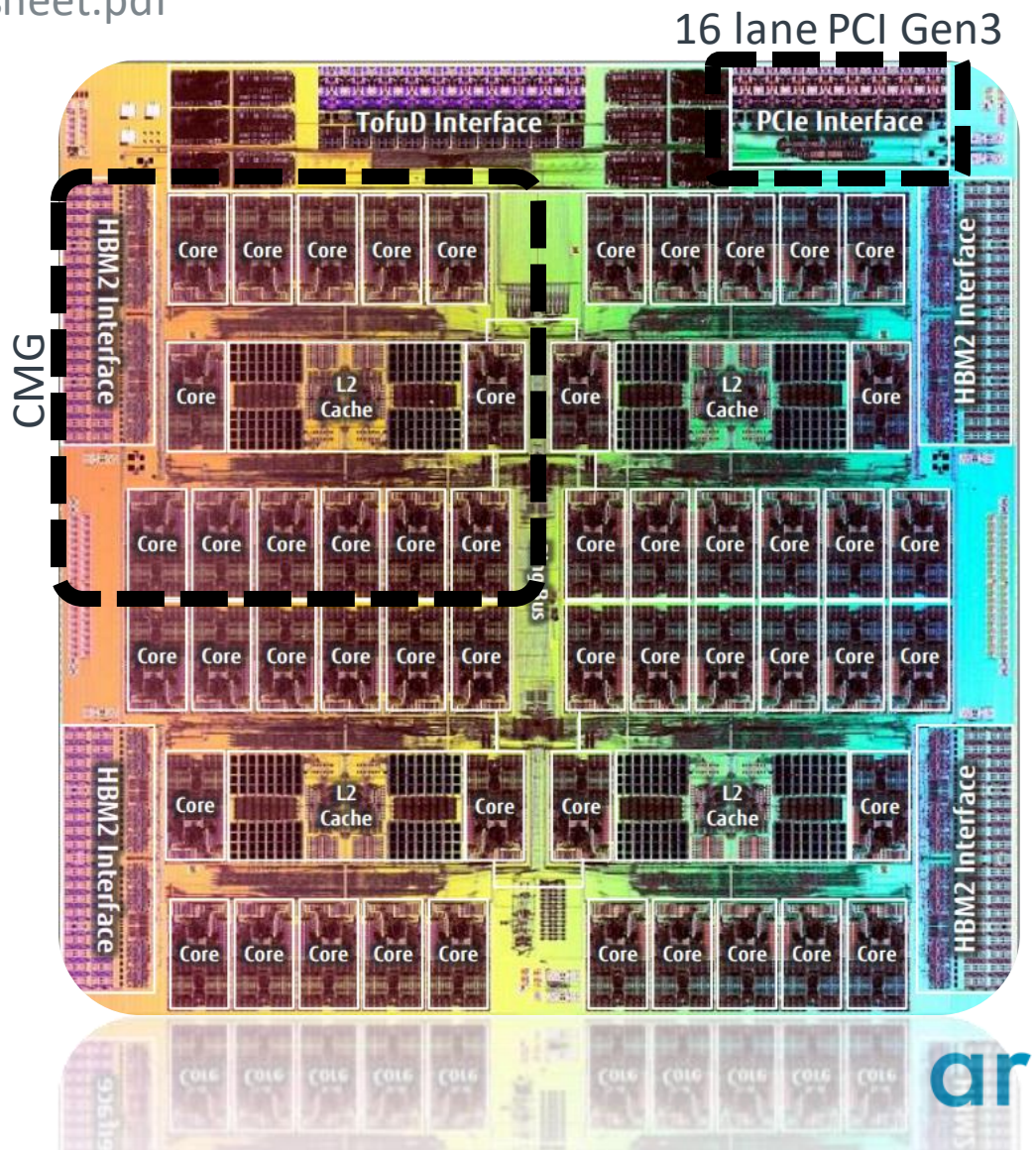
arm

Fujitsu A64FX Key Features

Architecture License

https://www.fujitsu.com/downloads/SUPER/a64fx/a64fx_datasheet.pdf

- Arm v8.2-A with 512-bit SVE
- Custom Fujitsu u-arch
- 7nm CMOS FinFET
- 2.2GHz, 2.0GHz, 1.8GHz
  - Constant clock: no turbo, no downclock
- **4 Core Memory Groups (CMGs)**
  - 12 cores (13 in the FX1000)
  - 64KB L1$ per core
    - 256b cache line
  - 8MB L2$ shared between all cores
    - 256b cache line
  - Zero L3$
  - 8 GB HBM at 256GB/s



arm

Example: FMLA

06_A64FX/01_fmla

See README.md for details

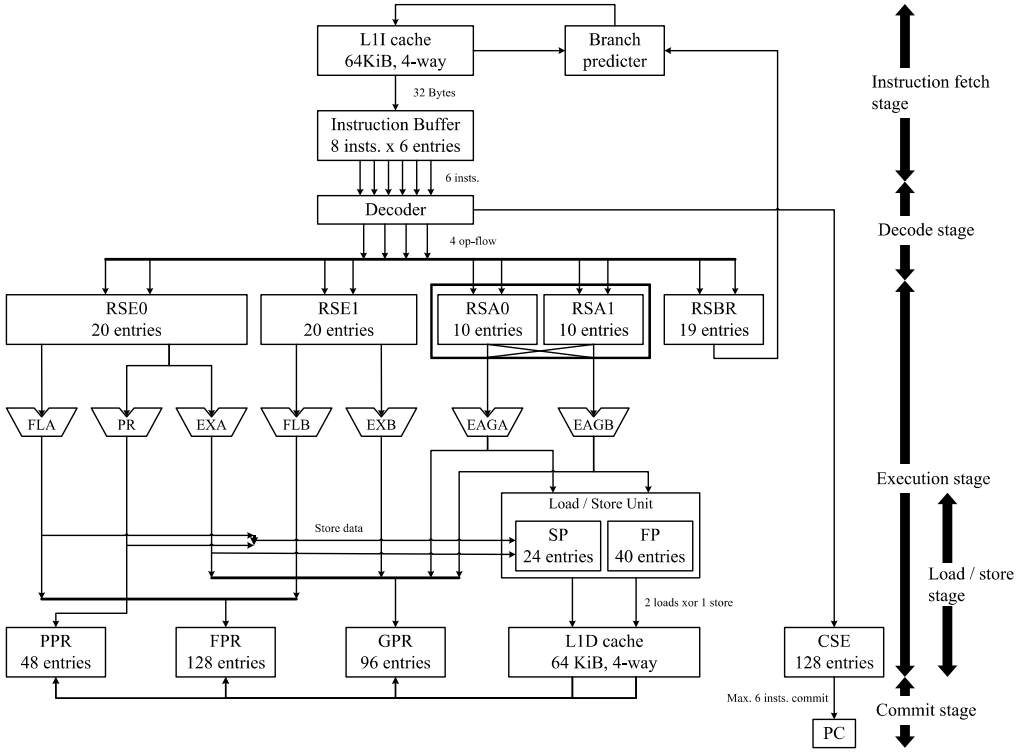
- Calculates peak per-core double precision flops
- Measures the wallclock time of a tight loop of fused multiply-add (FMLA) instructions.
- The code is written in Assembly, so the exact number of giga operations (GOP) is known.
- Performance in gigaflops (GFLOPS) is simply GFLOPS = GOP / SECONDS.

```
----------------------------------------
./fmla_neon128.exe
256000000 Flops in 0.021625 seconds
11.8382 GFlops
----------------------------------------
./fmla_sve512.exe
1024000000 Flops in 0.021627 seconds
47.3482 GFlops
----------------------------------------
./fmla_a64fx.exe
960000000 Flops in 0.016691 seconds
57.516 GFlops
----------------------------------------
```

wget https://gitlab.com/arm-hpc/training/arm-sve-tools/-/archive/sc20/arm-sve-tools-sc20.tar.gz

Fujitsu A64FX Execution Pipeline

https://github.com/fujitsu/A64FX/tree/master/doc



Fujitsu A64FX Execution Pipeline

https://github.com/fujitsu/A64FX/tree/master/doc

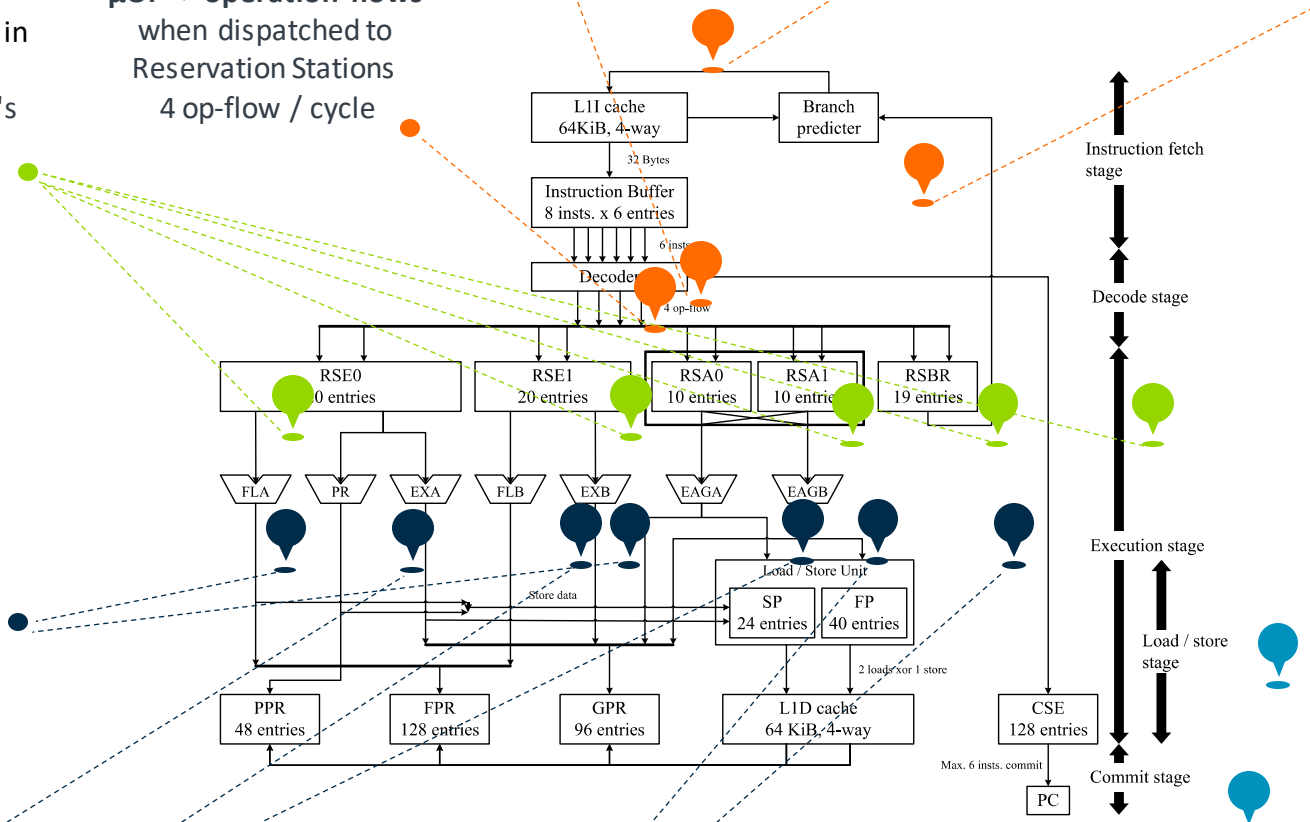
5 Reservation Stations (**RS**)
Each connected to a different execution pipeline
Operation-flows wait in RSs until the source operands are ready
A RS schedules op-flows to be executed in the pipelines
Only RSA0 and RSA1 can use each other's pipeline (share same issue port)
RSE0 cannot issue instructions to the FP pipeline attached to RSE1 (example)
Number of entries ↔ OoO resources

Architecture instructions decoded into µOP instructions (internal format)

µOP -> operation-flows when dispatched to Reservation Stations
4 op-flow / cycle

Where **architectural instruction** (assembly) to be executed is stored

Instruction fetch unit
up to 8 instructions / cycle from L1I cache
4 (up to 6) instructions / cycle to the decoder



5 functional stages

FP, SIMD & SVE pipelines
→2 FP instruction / cycle
Both supports FMA instructions
Only FLA does FP division

Predication pipeline

Integer pipelines

Address calculation pipelines
Generates addresses for loads and stores

Can execute 2 loads OR 1 store simultaneously

Check completion of instructions
Update state when all µOP of 1 instruction done

arm

Example: Stream

06_A64FX/02_stream/**01_stream_vanilla**

See README.md for details

- A basic, untuned, out-of-box, "vanilla" implementation
  - Performance will most likely be very poor
  - Uses only a single core and does not consider NUMA or any architectural features

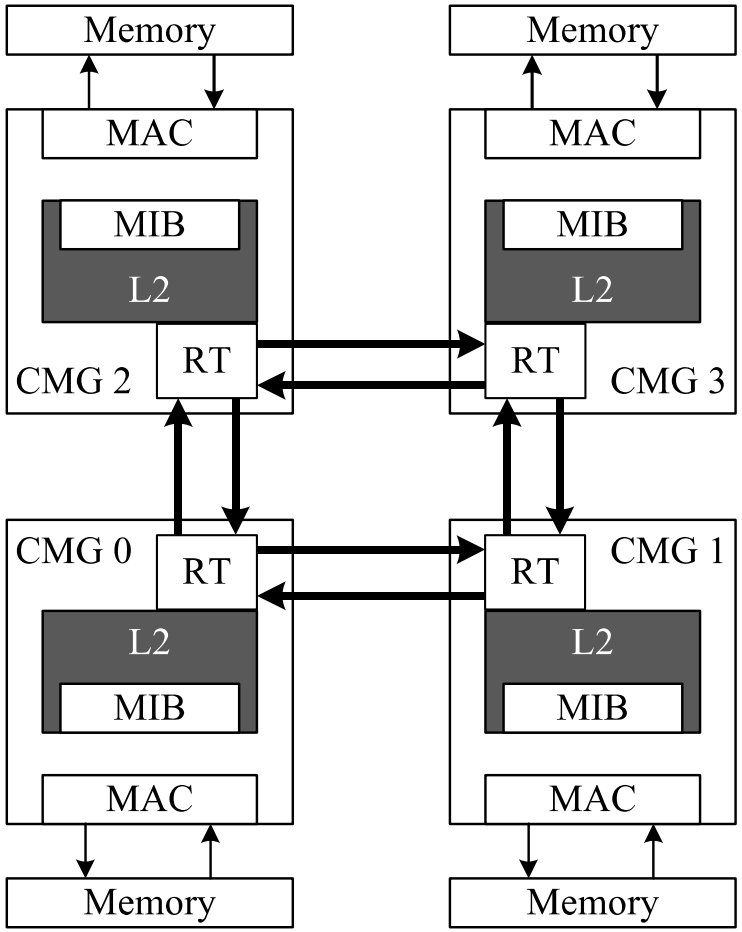
GCC 11 on A64FX

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           40859.3     0.003931     0.003916     0.003981
Scale:          40796.5     0.003931     0.003922     0.003949
Add:            47235.1     0.005109     0.005081     0.005188
Triad:          47253.3     0.005096     0.005079     0.005114
-------------------------------------------------------------
```

wget https://gitlab.com/arm-hpc/training/arm-sve-tools/-/archive/sc20/arm-sve-tools-sc20.tar.gz

Fujitsu A64FX L1 Cache

https://github.com/fujitsu/A64FX/tree/master/doc

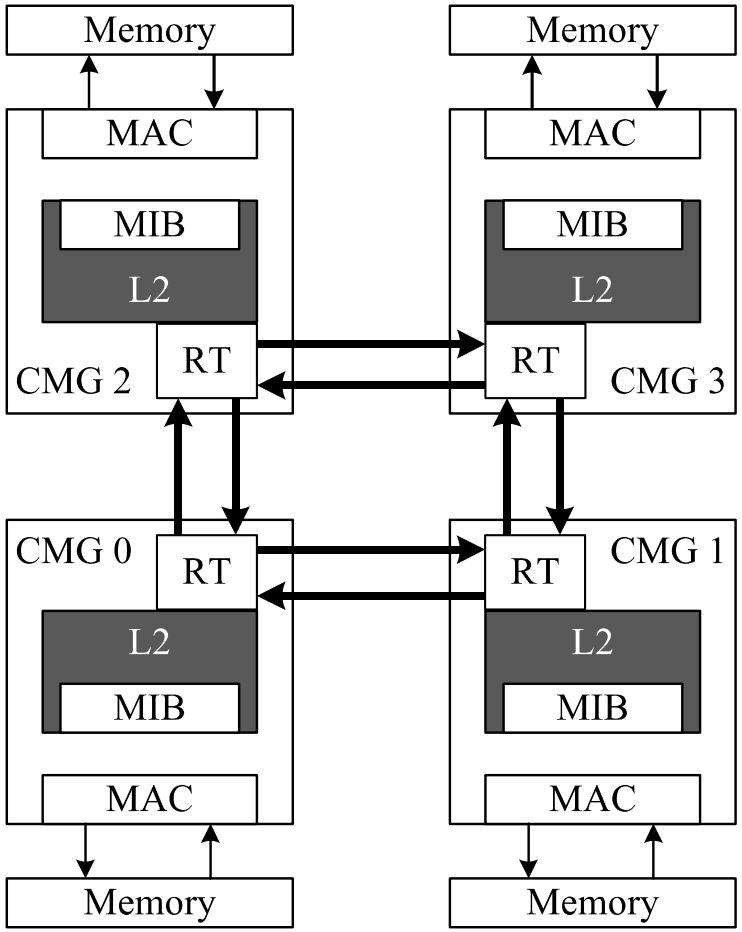


		For Instruction	For Data
L1 cache	Association method	4-way set associative	4-way set associative
	Capacity	64 KiB	64 KiB
	Hit latency (load-to-use)	4 cycles	5 cycles(integer)
			8 cycles (SIMD&FP / SVE in short mode)
			11 cycles (SIMD&FP / SVE in long mode)
	Line size	256 bytes	256 bytes
	Write method	---	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

arm

Fujitsu A64FX L2 Cache

https://github.com/fujitsu/A64FX/tree/master/doc



		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)	37 to 47 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

06_A64FX/02_stream/**02_stream_openmp**

See README.md for details

- Uses OpenMP and numactl to improve memory/thread locality
  - On many systems, this implementation will be close to 80% of the theoretical peak bandwidth
  - Does not achieve 80% of peak on A64FX due to that system's memory architecture

GCC 11 on A64FX

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           537948.0     0.032011     0.031936     0.032123
Scale:          537695.1     0.032026     0.031951     0.032179
Add:            597172.3     0.043259     0.043153     0.043500
Triad:          597324.1     0.043282     0.043142     0.044186
-------------------------------------------------------------
```

06_A64FX/02_stream/**04_stream_zfill**

See README.md for details

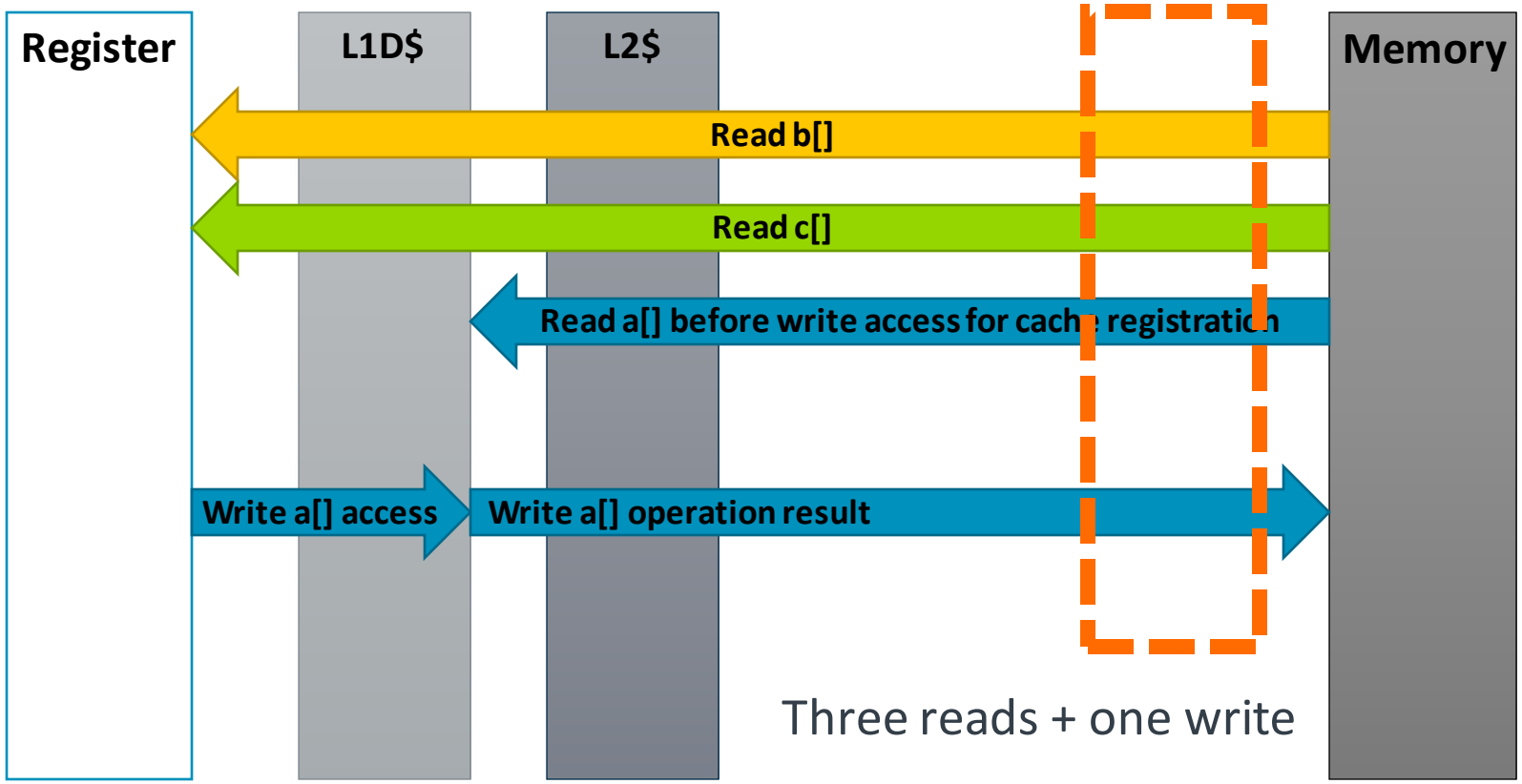
- Uses Arm's DC ZVA instruction to zero-fill cache lines
  - Dramatically improves the performance of systems with wide L2$ lines and low L3$

GCC 11 on A64FX

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           780579.1     0.022083     0.022009     0.022202
Scale:          780689.0     0.022146     0.022006     0.022576
Add:            788330.3     0.032902     0.032689     0.033698
Triad:          787974.0     0.032808     0.032704     0.033263
-------------------------------------------------------------
```

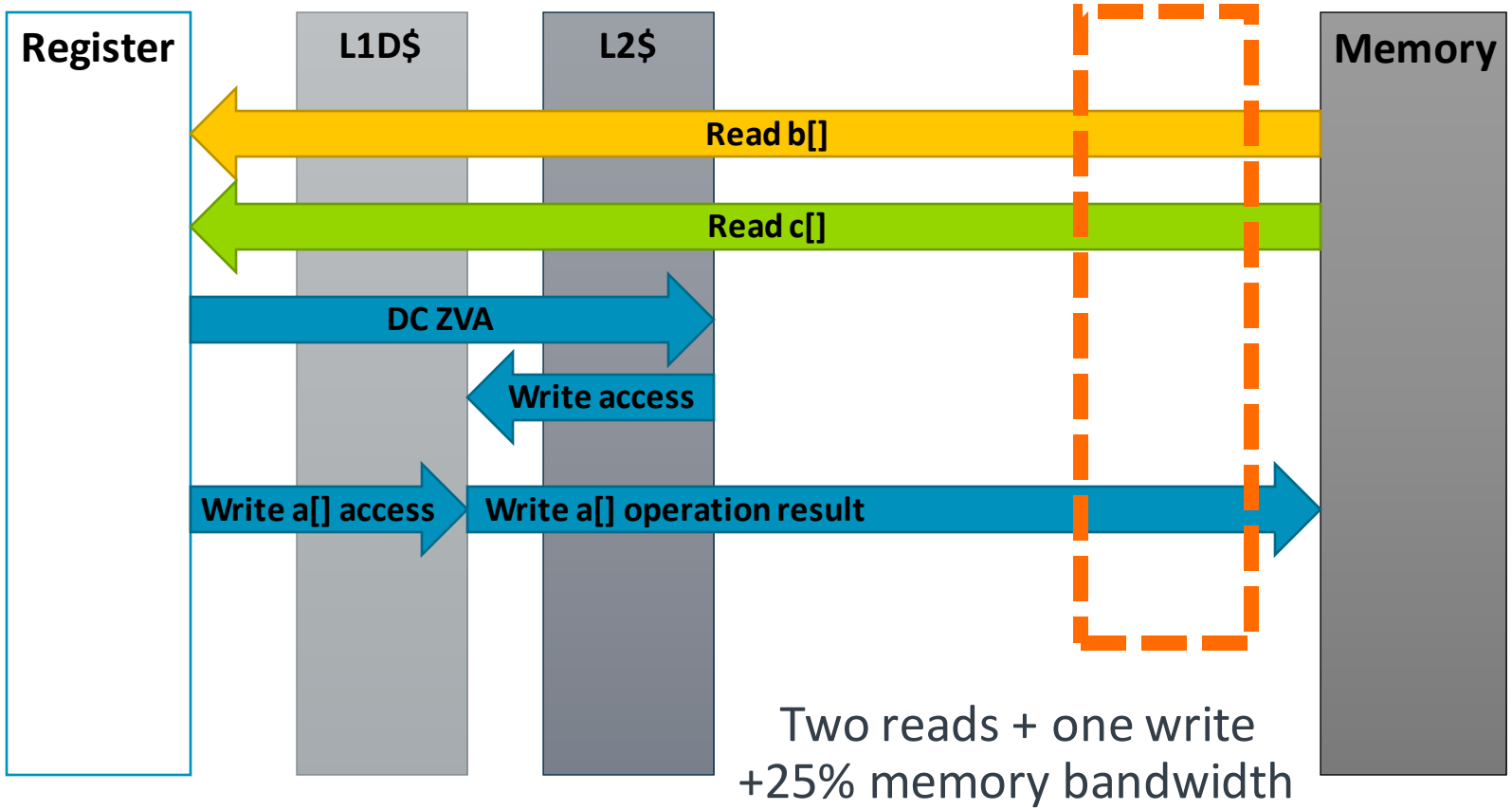
Unoptimized STREAM TRIAD

There is a hidden read before write of the result array a[] for cache registration



Use ZFILL to eliminate useless memory access

DC ZVA instruction maps cache without reading main memory



06_A64FX/02_stream/**05_stream_fujitsu**

See README.md for details

- Uses the Fujitsu compiler to maximize bandwidth on A64FX
  - No inline assembly
  - Compiler automatically inserts ZFILL instructions as needed

fcc 4.2.1 on Fujitsu A64FX

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           755455.3     0.022814     0.022741     0.022887
Scale:          768704.5     0.022393     0.022349     0.022464
Add:            819550.3     0.031496     0.031444     0.031545
Triad:          815555.5     0.031672     0.031598     0.031754
-------------------------------------------------------------
```

Resources

- Fujitsu A64FX Microarchitecture Manual
  - https://github.com/fujitsu/A64FX/tree/master/doc
- Fujitsu A64FX Performance Monitor Unit (PMU) events
  - https://github.com/fujitsu/A64FX/tree/master/doc
- Hands-on Training Materials for Arm Compilers, Libraries and Tools Related to SVE
  - https://gitlab.com/arm-hpc/training/arm-sve-tools
- Cray Apollo 80 Hardware Description
  - https://pubs.cray.com/bundle/HPE_Cray_Apollo_80_Hardware_Guide_H-6220/page/Product_Description.html
- Fujitsu PRIMEHPC Documentation
  - https://www.fujitsu.com/global/products/computing/servers/supercomputer/documents/

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה