



Olympus CPU Core Software Optimization Guide

Technical Reference

Version: 0.7

Date: 14-August-2025

ID: DP-12531-001

Table of Contents

1.	Introduction.....	4
1.1	Intended Audience.....	4
1.2	Scope.....	4
1.3	Glossary.....	4
1.4	Additional Reading.....	5
2.	About This Document	6
2.1	Identifying the Olympus core.....	6
3.	Instruction Characteristics	7
3.1	Understanding Instruction Tables.....	7
3.1.1	Description of Instruction Group and Instruction Columns	7
3.1.2	Description of Latency Column.....	7
3.1.3	Description of Throughput Column.....	8
3.1.4	Description of Utilized Pipelines Column	8
3.2	Branch Instructions.....	9
3.3	Arithmetic and Logical Instructions.....	10
3.4	Divide and Multiply Instructions.....	12
3.5	Pointer Authentication Instructions	12
3.6	Miscellaneous Data-Processing Instructions.....	14
3.7	Load Instructions	15
3.8	Store Instructions	17
3.9	Scalar/SIMD Floating-Point Instructions.....	18
3.10	SIMD FP8 Instructions.....	28
3.11	SIMD BF16 Instructions	28
3.12	SIMD Integer Instructions.....	29
3.13	Cryptography Extensions Instructions	38
3.14	3.15 FP Load Instructions.....	39
3.15	FP Store Instructions.....	39
3.16	SIMD Load Instructions.....	40
3.17	SIMD Store Instructions.....	41

3.18	SVE Predicate Instructions	43
3.19	SVE Floating-Point Instructions.....	46
3.20	SVE FP8 Instructions.....	51
3.21	SVE BF16 Instructions.....	52
3.22	SVE Integer Instructions.....	53
3.23	SVE Cryptography Instructions.....	70
3.24	SVE Load Instructions.....	71
3.25	SVE Store Instructions.....	78
4.	Special Considerations.....	82
4.1	Dispatch Constraints.....	82

1. Introduction

1.1 Intended Audience

The intended readers of this document are:

- Software developers using the Olympus CPU, in the areas of applications, OS, VMM, and firmware.
- Developers undertaking Performance tuning and CPU-related software debugging.

1.2 Scope

This document describes the micro-architecture aspects of the Olympus CPU core that influence software performance. Such Micro-architectural detail is limited to that useful for software optimization. The hardware rationale behind the software visible behavior and performance is not directly included.

1.3 Glossary

This document uses the following terms and abbreviations.

Term	Meaning
ALU	Arithmetic and Logical Unit
ASIMD	Advanced SIMD
MOp	Macro-Operation
μOp	Micro-Operation
SQRT	Square Root
FP	Floating-Point

1.4 Additional Reading

For additional information on Arm® v9 and functional description of the instructions described, see the Arm Architecture Reference Manual for A-profile architecture (<https://developer.arm.com/documentation/ddi0487>).

2. About This Document

The Olympus core is a high-performance and low-power core used in the Vera CPU. It implements the Arm v9.2-A architecture, and supports AArch64 execution state at all exception level (EL0 to EL3).

NOTE:

Cryptographic Extension support is SKU dependent, please refer to the datasheet of the exact SKU for more information. Further in this document, full support is assumed.

2.1 Identifying the Olympus core

The Olympus CPU core can be identified using the MIDR_EL1 register with the fields and Reset values as listed below:

Field	Value
Implementer	'N' (0x4e)
PartNum	0x10
Variant	
Revision	

The values in the Variant and Revision fields are used for detecting specific revisions of the core, not general Software optimization purposes.

3. Instruction Characteristics

This chapter describes the high-level performance characteristics for most Arm v9-A instructions. A series of tables summarize the effective execution latency, throughput, and pipelines utilized by each group of instructions.

3.1 Understanding Instruction Tables

In the Olympus core, instructions are first fetched and decoded into internal MOps. Those MOps are then renamed and dispatched to the out-of-order portion of the core. A MOp can be split into two μ Ops further down the pipeline after instruction decode. Once dispatched, a μ Op waits for its operands to become available and issues out-of-order to one of many execution pipelines. Each execution pipeline can accept one μ Op per cycle.

3.1.1 Description of Instruction Group and Instruction Columns

In the tables, the names used in the "Instruction" column match the textual names of the instructions contained w/in the Arm v9 Architectural Reference Manual. The names used in the "Instruction Group" column are simply a means of organizing the data, and have no additional meaning.

3.1.2 Description of Latency Column

In the tables, "Exec Latency" is generally defined as the minimum latency seen by a younger instruction which is dependent on an instruction in the described group.

The latencies shown for branches are measured from the point the operand flags/register is produced to the point the branch can possibly detect a misprediction.

The latencies shown for Loads assume the memory access hits in the Level 1 Data Cache and represent the maximal latency to produce the value in any register written by the instruction.

The latencies shown for Stores are measured from the point an operand register referenced by the Store is produced to the latest time that an address or data value is presented to the Store pipeline. Once executed, Stores are buffered and committed in the background.

When a latency is specified as "N (M)" in the tables below, this means the affected instructions support late-forwarding of the accumulator operands (when specified) from similar μ Ops, allowing a typical sequence of accumulating μ Ops to issue one every M cycles. The latency of the affected instructions is N cycles for inputs to non-accumulating operands.

3.1.3 Description of Throughput Column

In the tables, "Exec Throughput" is defined as the maximum throughput (in instructions per cycle) of an instruction in the specified instruction group that can be achieved by the Olympus core microarchitecture.

In the VX pipelines, various sets of instructions block subsequent similar operations to the same pipeline for N-1 additional cycles, where N equals the number of number of compatible execution units divided by the execution throughput. These instruction sets are annotated with the text "multi-pumped" in the Notes column.

The bandwidth of some more complex instructions is limited by decode constraints. These instructions are guaranteed to be decoded across multiple dispatch cycles, and may artificially limit the number of other instructions that can be dispatched together with the last dispatch cycle of the given instruction. Instructions affected by these decode constraints have "decoder line break" included in the Notes column. When this is dependent on the inclusion of a pre/post-increment Writeback, this text is parenthesized, similar to the "(I)" notation from the Utilized Pipelines column as described below.

When two threads are active in a core, the throughput numbers in the tables below are effectively halved for a given thread. The exception to this are when the 'M0' or 'V0' pipelines are utilized. Those pipelines are dedicated per-thread. In addition, latencies may be increased in cases where the halving of the throughput results in throughput values less than 1, since multiple μ Ops may be ready and wanting to issue to the same pipeline at the same time, but in this case, μ Ops may be delayed from starting execution due to this congestion.

3.1.4 Description of Utilized Pipelines Column

The execution pipelines used by an instruction's μ Op(s) are represented by the below symbols in the Utilized Pipelines column of the instruction tables:

Pipeline Types Table

Symbol	Pipeline
I	Integer single cycle 0/1/2/3/4/5 and single/multicycle 0/1.
M	Multicycle 0/1.
M0	Multicycle 0.

Symbol	Pipeline
B	Branch 0/1/2/3.
L	Load 0/1/2/3.
SA	Store 0/1.
D	Integer Store data 0/1.
V	FP/ASIMD 0/1/2/3/4/5.
V0123	FP/ASIMD 0/1/2/3 (also used for vector Store data).
V03	FP/ASIMD 0/3.
V12	FP/ASIMD 1/2.
V45	FP/ASIMD 4/5.
V0	FP/ASIMD 0.

Many memory operations have an optional pre/post-increment Writeback. When optional, this is indicated by a "(I)" included in the Utilized Pipelines column in the tables below.

3.2 Branch Instructions

AArch64 Branch Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Branch (simple)	B.cond	1	4	B	
	B				
	BR				
	RET				
Branch and Link	BL	1	4	I, B	
	BLR				
Compare and Branch	CBNZ	1	4	B	
	CBZ				
	TBNZ				
	TBZ				

3.3 Arithmetic and Logical Instructions

AArch64 Arithmetic and Logical Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Arithmetic / Logical (basic)	ADD (immediate)	1	8	1	
	ADD (shifted register)				
	ADD (extended register)				
	ADC				
	AND (immediate)				
	AND (shifted register)				
	BIC (shifted register)				
	EON (shifted register)				
	EOR (immediate)				
	EOR (shifted register)				
	GMI				
	ORN (shifted register)				
	ORR (immediate)				
	ORR (shifted register)				
	SUB (immediate)				
	SUB (shifted register)				
	SUB (extended register)				
	SUBP				
	SBC				
Arithmetic / Logical (basic, flag Write)	ADDS (immediate)	1	6	1	
	ADDS (shifted register)				
	ADDS (extended register)				
	ADCS				
	ANDS (immediate)				
	ANDS (shifted register)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	BICS (shifted register)				
	SUBS (immediate)				
	SUBS (shifted register)				
	SUBS (extended register)				
	SUBPS				
	SBCS				
Arithmetic (src extend from H/B)	ADD (extended register)				
	ADDS (extended register)	2	2	M	
	SUB (extended register)				
	SUBS (extended register)				
Arithmetic (src LSL Shift > 4 or LSR/ASR/ ROR Shift)	ADD (shifted register)				
	ADDS (shifted register)	2	2	M	
	SUB (shifted register)				
	SUBS (shifted register)				
Logical (flag Write, src Shift)	ANDS (shifted register)	2	2	M	
	BICS (shifted register)				
Conditional Select	CSEL				
	CSINC	1	8	I	
	CSINV				
	CSNEG				
Conditional Compare	CCMN (immediate)				
	CCMP (immediate)	1	6	I	
	CCMN (register)				
	CCMP (register)				
Flag manipulation	SETF8, SETF16				
	RMIF	1	6	I	
	CFINV				
	AXFLAG				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	XAFLAG				
Arithmetic to Tag	ADDG	2	2	M	
	SUBG				
Insert Random Tag (GCR_EL1.RRND=1)	IRG	2	2	M	decoder line break
Insert Random Tag (GCR_EL1.RRND=0)	IRG	5	0.5	M, 2*M0	decoder line break

3.4 Divide and Multiply Instructions

AArch64 Divide and Multiply Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Divide	SDIV	5 (variable)	2	M	Divides are performed using an iterative algorithm and block any subsequent divide operations until complete. Early termination is possible, depending on the data values. Latencies are up to 12 cycles for the W form and up to 20 cycles for the X form.
	UDIV				
Multiply accumulate	MADD	2 (1)	2	M	
	MSUB				
	SMADDL				
	UMADDL				
	SMSUBL				
	UMSUBL				
Multiply high	SMULH	3	2	M	
	UMULH				

3.5 Pointer Authentication Instructions

AArch64 Pointer Authentication Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Authenticate address	AUTDA AUTDZA	4	1	M0	
	AUTDB AUTDZB				
	AUTIA AUTIA1716 AUTIASP AUTIAZ AUTIZA				
	AUTIB AUTIB1716 AUTIBSP AUTIBZ AUTIZB				
Compute pointer authentication code	PACDA PACDZA	4	1	M0	
	PACDB PACDZB				
	PACGA				
	PACIA PACIA1716 PACIASP PACIAZ PACIZA				
	PACIB PACIB1716 PACIBSP PACIBZ PACIZB				
Strip pointer authentication code	XPACD XPACI XPACLRI	2	1	M0	
Branch, register with authentication	BRAA BRAAZ BRAB BRABZ	5	1	B, M0	
	RETA RETAB				
Branch and link, register with authentication	BLRAA BLRAAZ BLRAB BLRABZ	5	1	I, B, M0	
Load, register with authentication	LDRAA LDRAB	8	1	M0, L, (I)	

3.6 Miscellaneous Data-Processing Instructions

AArch64 Miscellaneous Data-Processing Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Address generation	ADR	1	6	I	
	ADRP				
Extract, ROR alias or (imms==0)	EXTR	1	8	I	
Extract, other	EXTR	3	2	I, M	
Bitfield move, basic	SBFM	1	8	I	
	UBFM				
Bitfield move, insert	BFM	2	2	M	
Count leading	CLS	1	8	I	
	CLZ				
Move imm	MOVK	1	8	I	
	MOVN				
	MOVZ				
Reverse bits/bytes	RBIT	1	8	I	
	REV				
	REV16				
	REV32				
Variable Shift	ASRV	1	8	I	
	LSLV				
	LSRV				
	RORV				
CRC checksum	CRC32B, CRC32H, CRC32W, CRC32X	2 (1)	2	M	
	CRC32CB, CRC32CH, CRC32CW, CRC32CX				

3.7 Load Instructions

AArch64 Load Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load register, possible wback	LDR (immediate)	4	4	L, (I)	
	LDRB (immediate)				
	LDRH (immediate)				
	LDRSB (immediate)				
	LDRSH (immediate)				
	LDRSW (immediate)				
Load register, bare	LDR (register)	4	4	L	
	LDRB (register)				
	LDRH (register)				
	LDRSB (register)				
	LDRSH (register)				
	LDRSW (register)				
	LDUR				
	LDURB				
	LDURH				
	LDURSB				
	LDURSH				
	LDURSW				
	LDAR				
	LDARB				
	LDARH				
	LDAPR				
	LDAPRB				
	LDAPRH				
LDAPUR					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LDAPURB				
	LDAPURH				
	LDAPURSB				
	LDAPURSH				
	LDAPURSW				
	LDLAR				
	LDLARB				
	LDLARH				
	LDTR				
	LDTRB				
	LDTRH				
	LDTRSB				
	LDTRSH				
	LDTRSW				
	LDXR				
	LDXRB				
	LDXRH				
	LDAXR				
	LDAXRB				
	LDAXRH				
Load pair, possible wback	LDP	4	4	L, (I)	
	LDPSW				
Load pair, bare	LDNP	4	4	L	
	LDXP				
	LDAXP				
Load literal	LDR (literal)	5	4	I, L	
	LDRSW (literal)				
	PRFM (literal)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load allocation tag	LDG	5	4	I, L	

3.8 Store Instructions

AArch64 Store Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store register, possible wback	STR (immediate)	1	2	SA, D, (I)	
	STRB (immediate)				
	STRH (immediate)				
Store register, scaled by 2	STRH (register)	2	2	I, SA, D	
Store register, bare	STR (register)	1	2	SA, D	
	STRH (register)				
	STRB (register)				
	STUR				
	STURB				
	STURH				
	STLR				
	STLRB				
	STLRH				
	STLLR				
	STLLRB				
	STLLRH				
	STLUR				
	STLURB				
	STLURH				
STTR					
STTRB					
STTRH					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store pair, gen register	STP	1	2	SA, D, (I)	
Store pair, non-temporal	STNP	1	2	SA, D	
Store exclusive	STXR	4	2	SA, D	
	STXRB				
	STXRH				
	STLXR				
	STLXRB				
	STLXRH				
	STXP				
	STLXP				
Store allocation tag	ST2G	1	2	SA, D, (I)	
	STG				
	STZ2G				
	STZG				
	STGP				

3.9 Scalar/SIMD Floating-Point Instructions

AArch64 Scalar/SIMD Floating-Point Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	FMOV (scalar, immediate)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP misc	FMOV (register)	2	6	V	
	FMOV (vector, immediate)				
	FCSEL				
	FABS (scalar)				
	FABD				
	FABS (vector)				
	FNEG (scalar)				
	FNEG (vector)				
	FADD (scalar)				
	FADD (vector)				
	FADDP (scalar)				
	FADDP (vector)				
	FCADD				
	FSUB (scalar)				
	FSUB (vector)				
	FACGE				
	FACGT				
	FCMEQ (register)				
	FCMGE (register)				
	FCMGT (register)				
	FCMEQ (zero)				
	FCMGE (zero)				
	FCMGT (zero)				
	FCMLE (zero)				
	FCMLT (zero)				
	FMAX (vector)				
	FMAXNM (vector)				
	FMAX (scalar)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	FMAXNM (scalar)				
	FMAXNMP (scalar)				
	FMAXP (scalar)				
	FMAXNMP (vector)				
	FMAXP (vector)				
	FMIN (vector)				
	FMINNM (vector)				
	FMIN (scalar)				
	FMINNM (scalar)				
	FMINNMP (scalar)				
	FMINP (scalar)				
	FMINNMP (vector)				
	FMINP (vector)				
	FAMAX				
	FAMIN				
FP compare	FCCMP	2	2	V03	
	FCCMPE				
	FCMP				
	FCMPE				
FP multiply	FMUL (scalar)	3	6	V	FP multiply accumulate pipelines support late forwarding of the result from FP multiply μ Ops to the accumulate operand. The latency between the issue of the multiply producer and the multiply accumulate consumer is 2 cycles in this case.
	FNMUL (scalar)				
	FMUL (by element)				
	FMULX (by element)				
	FMUL (vector)				
	FMULX				
	FSCALE				
	FMADD				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP multiply accumulate	FMADD	4 (2)	6	V	
	FMSUB				
	FNMSUB				
	FCMLA (by element)				
	FCMLA				
	FMLA (by element)				
	FMLAL, FMLAL2 (by element)				
	FMLS (by element)				
	FMLSL, FMLSL2 (by element)				
	FMLA (vector)				
	FMLAL, FMLAL2 (vector)				
	FMLS (vector)				
	FMLSL, FMLSL2 (vector)				
FP reduction, DP/SP/(HP 64b)	FMAXNMV	4	3	2*V	
	FMAXV				
	FMAXNMV				
	FMAXV				
	FMINNMV				
	FMINV				
	FMINNMV				
	FMINV				
FP reduction, HP 128b	FMAXNMV	6	2	3*V	decoder line break
	FMAXV				
	FMINNMV				
	FMINV				
FP divide/sqrt, scalar DP	FDIV (scalar)	13	2	V12	
	FSQRT (scalar)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP divide/sqrt, scalar SP	FDIV (scalar)	8	2	V12	
	FSQRT (scalar)				
FP divide/sqrt, scalar HP	FDIV (scalar)	6	2	V12	
	FSQRT (scalar)				
FP divide/sqrt, vector DP 128b	FDIV (vector)	14	1	V12	multi-pumped
	FSQRT (vector)				
FP divide/sqrt, vector SP 128b	FDIV (vector)	11	0.5	V12	multi-pumped
	FSQRT (vector)				
FP divide/sqrt, vector SP 64b	FDIV (vector)	9	1	V12	multi-pumped
	FSQRT (vector)				
FP divide/sqrt, vector HP 128b	FDIV (vector)	13	0.25	V12	multi-pumped
	FSQRT (vector)				
FP divide/sqrt, vector HP 64b	FDIV (vector)	9	0.5	V12	multi-pumped
	FSQRT (vector)				
FP round to int, scalar	FRINTA (scalar)	3	4	V0123	
	FRINTI (scalar)				
	FRINTM (scalar)				
	FRINTN (scalar)				
	FRINTP (scalar)				
	FRINTX (scalar)				
	FRINTZ (scalar)				
	FRINT32X (scalar)				
	FRINT32Z (scalar)				
	FRINT64X (scalar)				
	FRINT64Z (scalar)				
	FRINTA (vector)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP round to int, vector, DP 128b, or SP 64b	FRINTI (vector)	3	4	V0123	
	FRINTM (vector)				
	FRINTN (vector)				
	FRINTP (vector)				
	FRINTX (vector)				
	FRINTZ (vector)				
	FRINT32X (vector)				
	FRINT32Z (vector)				
	FRINT64X (vector)				
	FRINT64Z (vector)				
FP round to int, vector, SP 128b, or HP 64b	FRINTA (vector)	4	2	V0123	multi-pumped
	FRINTI (vector)				
	FRINTM (vector)				
	FRINTN (vector)				
	FRINTP (vector)				
	FRINTX (vector)				
	FRINTZ (vector)				
	FRINT32X (vector)				
	FRINT32Z (vector)				
	FRINT64X (vector)				
FRINT64Z (vector)					
FP round to int, vector, HP 128b	FRINTA (vector)	6	1	V0123	multi-pumped
	FRINTI (vector)				
	FRINTM (vector)				
	FRINTN (vector)				
	FRINTP (vector)				
	FRINTX (vector)				
	FRINTZ (vector)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP convert (scalar)	FCVT	3	4	V0123	
	FCVTAS (vector)				
	FCVTAU (vector)				
	FCVTMS (vector)				
	FCVTMU (vector)				
	FCVTNS (vector)				
	FCVTNU (vector)				
	FCVTPS (vector)				
	FCVTPU (vector)				
	FCVTXN, FCVTXN2				
	FCVTZS (vector, integer)				
	FCVTZU (vector, integer)				
	FCVTZS (vector, fixed-point)				
	FCVTZU (vector, fixed-point)				
	SCVTF (vector, integer)				
	UCVTF (vector, integer)				
	SCVTF (vector, fixed-point)				
UCVTF (vector, fixed-point)					
FP convert (vector between F32 and F64)	FCVTL, FCVTL2	3	4	V0123	
	FCVTN, FCVTN2 (FP64 to FP32)				
FP convert (vector F32 to F16)	FCVTN, FCVTN2 (FP32 to FP16)	4	4	V0123	
FP convert (vector F16 to F32)	FCVTL, FCVTL2	4	2	V0123	multi-pumped
FP convert (java script)	FJCVTZS	4	2	V03	
	FCVTAS (vector)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP convert (vector, DP 128b or SP 64b)	FCVTAU (vector)	3	4	V0123	
	FCVTMS (vector)				
	FCVTMU (vector)				
	FCVTNS (vector)				
	FCVTNU (vector)				
	FCVTPS (vector)				
	FCVTPU (vector)				
	FCVTZS (vector, integer)				
	FCVTZU (vector, integer)				
	FCVTZS (vector, fixed-point)				
	FCVTZU (vector, fixed-point)				
	SCVTF (vector, integer)				
	UCVTF (vector, integer)				
	SCVTF (vector, fixed-point)				
	UCVTF (vector, fixed-point)				
FCVTXN, FCVTXN2					
FP convert (vector, SP 128b or HP 64b)	FCVTAS (vector)	4	2	V0123	multi-pumped
	FCVTAU (vector)				
	FCVTMS (vector)				
	FCVTMU (vector)				
	FCVTNS (vector)				
	FCVTNU (vector)				
	FCVTPS (vector)				
	FCVTPU (vector)				
	FCVTZS (vector, integer)				
	FCVTZU (vector, integer)				
	FCVTZS (vector, fixed-point)				
	FCVTZU (vector, fixed-point)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SCVTF (vector, integer)				
	UCVTF (vector, integer)				
	SCVTF (vector, fixed-point)				
	UCVTF (vector, fixed-point)				
FP convert (vector, HP 128b)	FCVTAS (vector)	6	1	V0123	multi-pumped
	FCVTAU (vector)				
	FCVTMS (vector)				
	FCVTMU (vector)				
	FCVTNS (vector)				
	FCVTNU (vector)				
	FCVTPS (vector)				
	FCVTPU (vector)				
	FCVTZS (vector, integer)				
	FCVTZU (vector, integer)				
	FCVTZS (vector, fixed-point)				
	FCVTZU (vector, fixed-point)				
	SCVTF (vector, integer)				
	UCVTF (vector, integer)				
	SCVTF (vector, fixed-point)				
	UCVTF (vector, fixed-point)				
FP convert (from gen reg to vector reg)	SCVTF (scalar, integer)	5	2	M, V	
	UCVTF (scalar, integer)				
	SCVTF (scalar, fixed-point)				
	UCVTF (scalar, fixed-point)				
	FCVTAS (scalar)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP convert (from vector reg to gen reg)	FCVTAU (scalar)	4	2	V03	
	FCVTMS (scalar)				
	FCVTMU (scalar)				
	FCVTNS (scalar)				
	FCVTNU (scalar)				
	FCVTPS (scalar)				
	FCVTPU (scalar)				
	FCVTZS (scalar, fixed-point)				
	FCVTZU (scalar, fixed-point)				
	FCVTZS (scalar, integer)				
	FCVTZU (scalar, integer)				
FP reciprocal and square root estimate (scalar)	FRECPX	3	4	V0123	
	FRECPE				
	FRSQRTE				
FP reciprocal and square root estimate (vector, DP 128b or SP 64b)	FRECPE	3	4	V0123	
	FRSQRTE				
FP reciprocal and square root estimate (vector, SP 128b or HP 64b)	FRECPE	4	2	V0123	multi-pumped
	FRSQRTE				
FP reciprocal and square root estimate (vector, HP 128b)	FRECPE	6	1	V0123	multi-pumped
	FRSQRTE				
FP reciprocal and square root step	FRECPS	4 (2)	6	V	
	FRSQRTS				
FP mov (from vector reg to gen reg)	FMOV (general)	3	2	V03	
FP mov (from gen reg to low half of vector reg)	FMOV (general)	3	2	M	
FP mov (from gen reg to high half of vector reg)	FMOV (general)	5	2	M, V	

3.10 SIMD FP8 Instructions

AArch64 SIMD FP8 Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Convert to/from F16	F1CVTL, F1CVTL2, F2CVTL, F2CVTL2	4	2	V0123	multi-pumped
	FCVTN (FP16 to FP8)				
Convert from F32	FCVTN, FCVTN2 (FP32 to FP8)	3	4	V0123	
Dot product, Multiply accumulate	FDOT (2-way, vector)	4 (2)	6	V	
	FDOT (4-way, vector)				
	FDOT (2-way, by element)				
	FDOT (4-way, by element)				
	FMLALB, FMLALT (vector)				
	FMLALLBB, FMLALLBT, FMLALLTB, FMLALLTT (vector)				
	FMLALB, FMLALT (by element)				
	FMLALLBB, FMLALLBT, FMLALLTB, FMLALLTT (by element)				

3.11 SIMD BF16 Instructions

AArch64 SIMD BF16 Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Convert from F8	BF1CVTL, BF1CVTL2, BF2CVTL, BF2CVTL2	4	2	V0123	multi-pumped
Convert from F32 (vector)	BFCVTN, BFCVTN2	4	2	V0123	multi-pumped
Convert from F32 (scalar)	BFCVT	3	4	V0123	
Multiply accumulate	BFMLALB, BFMLALT (vector)	4 (2)	6	V	
	BFMLALB, BFMLALT (by element)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Dot product	BFDOT (vector)	5 (3)	6	V	
	BFDOT (by element)				
Matrix multiply accumulate	BFMMLA	6 (4)	6	V	

3.12 SIMD Integer Instructions

AArch64 SIMD Integer Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	ABS				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	ADD (vector)				
	ADDHN, ADDHN2				
	ADDP (scalar)				
	ADDP (vector)				
	AND (vector)				
	BIC (vector, immediate)				
	BIC (vector, register)				
	BIF				
	BIT				
	BSL				
	CLS (vector)				
	CLZ (vector)				
	CMEQ (register)				
	CMGE (register)	2	6	V	
	CMGT (register)				
	CMEQ (zero)				
	CMGE (zero)				
	CMGT (zero)				
	CMHI (register)				
	CMHS (register)				
	CMLE (zero)				
	CMLT (zero)				
	CMTST				
	CNT				
	DUP (element)				
	EOR (vector)				
	EXT				
	INS (element)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LUTI2				
	LUTI4				
	MOVI				
	MVNI				
	NEG (vector)				
	NOT				
	ORN (vector)				
	ORR (vector, immediate)				
	ORR (vector, register)				
	RADDHN, RADDHN2				
	RBIT (vector)				
	REV16 (vector)				
	REV32 (vector)				
	REV64				
	RSUBHN, RSUBHN2				
	SHL				
	SHLL, SHLL2				
	SHRN, SHRN2				
	SLI				
	SRI				
	SUB (vector)				
	SUBHN, SUBHN2				
	TRN1				
	TRN2				
	UZP1				
	UZP2				
	XTN, XTN2				
	ZIP1				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	ZIP2				
	FABD				
	SABD				
	SABDL, SABDL2				
	UABD				
	UABDL, UABDL2				
	SADDL, SADDL2				
	SADDLP				
	UADDL, UADDL2				
	UADDLP				
	SADDW, SADDW2				
	UADDW, UADDW2				
	SHADD				
	SRHADD				
	UHADD				
	URHADD				
	SHSUB				
	UHSUB				
	SMAX				
	SMAXP				
	UMAX				
	UMAXP				
	SMIN				
	SMINP				
	UMIN				
	UMINP				
	SQABS				
	SQADD				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	UQADD				
	SQNEG				
	SQSUB				
	UQSUB				
	SSHL				
	SSHLL, SSHLL2				
	USHL				
	USHLL, USHLL2				
	SSHR				
	USHR				
	SSUBL, SSUBL2				
	SSUBW, SSUBW2				
	USUBL, USUBL2				
	USUBW, USUBW2				
	SUQADD				
	USQADD				
Shift	RSHRN, RSHRN2	4	6	V	
	SQRSHL				
	UQRSHL				
	SQRSHRN, SQRSHRN2				
	SQRSHRUN, SQRSHRUN2				
	UQRSHRN, UQRSHRN2				
	SQSHL (immediate)				
	UQSHL (immediate)				
	SQSHL (register)				
	UQSHL (register)				
	SQSHLU				
	SQSHRN, SQSHRN2				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SQSHRUN, SQSHRUN2				
	UQSHRN, UQSHRN2				
	SRSHL				
	SRSHR				
	URSHL				
	URSHR				
	SQXTN, SQXTN2				
	SQXTUN, SQXTUN2				
	UQXTN, UQXTN2				
Multiply	MUL (vector)	4	4	V0123	
	MUL (by element)				
	SMULL, SMULL2 (vector)				
	UMULL, UMULL2 (vector)				
	SMULL, SMULL2 (by element)				
	UMULL, UMULL2 (by element)				
	SQDMULH (vector)				
	SQDMULL, SQDMULL2 (vector)				
	SQRDMULH (vector)				
	SQDMULH (by element)				
	SQDMULL, SQDMULL2 (by element)				
	SQRDMULH (by element)				
	MLA (vector)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Multiply accumulate	MLS (vector)	4 (2)	4	V0123	
	SMLAL, SMLAL2 (vector)				
	SMLS�, SMLS�2 (vector)				
	UMLAL, UMLAL2 (vector)				
	UMLS�, UMLS�2 (vector)				
	MLA (by element)				
	MLS (by element)				
	SMLAL, SMLAL2 (by element)				
	SMLS�, SMLS�2 (by element)				
	UMLAL, UMLAL2 (by element)				
	UMLS�, UMLS�2 (by element)				
	SQDMLAL, SQDMLAL2 (vector)				
	SQDMLS�, SQDMLS�2 (vector)				
	SQRDMLAH (vector)				
	SQRDMLS�H (vector)				
	SQDMLAL, SQDMLAL2 (by element)				
	SQDMLS�, SQDMLS�2 (by element)				
	SQRDMLAH (by element)				
	SQRDMLS�H (by element)				
Abs diff, Shift accumulate	SABA	4 (2)	6	V	
	SABAL, SABAL2				
	UABA				
	UABAL, UABAL2				
	SADALP				
	UADALP				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SRSRA				
	SSRA				
	URSRA				
	USRA				
Dot product, Matrix multiply	SDOT (vector)	3 (2)	6	V	
	UDOT (vector)				
	USDOT (vector)				
	SDOT (by element)				
	SUDOT (by element)				
	UDOT (by element)				
	USDOT (by element)				
	SMMLA (vector)				
	UMMLA (vector)				
	USMMLA (vector)				
Arithmetic reduce (4H/4S)	ADDV	3	4	V0123	
	SADDLV				
	UADDLV				
	SMAXV				
	UMAXV				
	SMINV				
	UMINV				
Arithmetic reduce (8B/8H)	ADDV	5	3	V, V0123	
	SADDLV				
	UADDLV				
	SMAXV				
	UMAXV				
	SMINV				
	UMINV				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Arithmetic reduce (16B)	ADDV	6	2	2*V0123	
	SADDLV				
	UADDLV				
	SMAXV				
	UMAXV				
	SMINV				
	UMINV				
Polynomial multiply	PMUL	3	6	V	
Polynomial multiply long	PMULL, PMULL2	2	6	V	This category only includes instructions with a destination format of 8H
Reciprocal and square root estimate (2S)	URECPE	3	4	V0123	
	URSQRTE				
Reciprocal and square root estimate (4S)	URECPE	4	2	V0123	multi-pumped
	URSQRTE				
Table lookup (single- or 2-reg table)	TBL	2	6	V	
Table lookup (3-reg table)	TBL	4	3	2*V	
Table lookup (4-reg table)	TBL	4	2	3*V	decoder line break
Table lookup extension (single-reg table)	TBX	2	6	V	
Table lookup extension (2-reg table)	TBX	4	3	2*V	
Table lookup extension (3-reg table)	TBX	6	2	3*V	decoder line break
Table lookup extension (4-reg table)	TBX	6	1.2	5*V	decoder line break
Transfer vector reg to gen reg	SMOV	3	2	V03	
	UMOV				
Transfer gen reg to vector reg	DUP (general)	3	2	M	
Insert gen reg to vector reg	INS (general)	5	2	M, V	

3.13 Cryptography Extensions Instructions

AArch64 Cryptography Extensions Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
AES, SHA3 operations	AESD	2	6	V	
	AESE				
	AESIMC				
	AESMC				
	BCAX				
	EOR3				
	RAX1				
	XAR				
Polynomial Multiply Long	PMULL, PMULL2	2	6	V	This category only includes instructions with a destination format of 1Q
SHA general	SHA1H	2	1	V0	
	SHA1SU0				
	SHA1SU1				
	SHA256SU0				
	SHA256SU1				
	SHA512H2				
	SHA512H				
	SHA512SU0				
	SHA512SU1				
SHA hash acceleration	SHA1C	4	1	V0	
	SHA1M				
	SHA1P				
	SHA256H2				
	SHA256H				
	SM3PARTW1				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
SM3 operations	SM3PARTW2	2	4	V0123	
	SM3SS1				
	SM3TT1A				
	SM3TT1B				
	SM3TT2A				
	SM3TT2B				
SM4 operations	SM4E	4	1	V0	
	SM4EKEY				

3.14 3.15 FP Load Instructions

AArch64 FP Load Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load vector register (immediate)	LDR (immediate, SIMD&FP)	6	4	L, (I)	
Load vector register (unscaled immediate)	LDUR (SIMD&FP)	6	4	L	
Load vector register (register)	LDR (register, SIMD&FP)	7	4	I, L	
Load vector register (literal)	LDR (literal, SIMD&FP)	7	4	I, L	
Load vector pair (128b)	LDP (SIMD&FP)	6	2	2*L, (I)	(decoder line break)
Load vector pair (32b/64b)	LDP (SIMD&FP)	6	4	L, (I)	
Load vector pair non-temporal (128b)	LDNP (SIMD&FP)	6	2	2*L	
Load vector pair non-temporal (32b/64b)	LDNP (SIMD&FP)	6	4	L	

3.15 FP Store Instructions

AArch64 FP Store Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store vector register (immediate)	STR (immediate, SIMD&FP)	3	2	SA, V0123, (I)	
Store vector register (unscaled immediate)	STUR (SIMD&FP)	3	2	SA, V0123	
Store vector register (register)	STR (register, SIMD&FP)	3	2	I, SA, V0123	
Store vector pair (128 bit)	STP (SIMD&FP)	3	1	SA, V0123, (I)	
Store vector pair (32/64 bit)	STP (SIMD&FP)	3	2	SA, V0123, (I)	
Store vector pair non-temporal (128 bit)	STNP (SIMD&FP)	3	1	SA, V0123	
Store vector pair non-temporal (32/64 bit)	STNP (SIMD&FP)	3	2	SA, V0123	

3.16 SIMD Load Instructions

AArch64 SIMD Load Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load multiple 1-element structures to 2 registers	LD1 (multiple structures)	6	4	L, (I)	
Load multiple 1-element structures to 2 registers (D form)	LD1 (multiple structures)	6	4	L, (I)	
Load multiple 1-element structures to 2 registers (Q form)	LD1 (multiple structures)	6	2	2*L, (I)	(decoder line break)
Load multiple 1-element structures to 3 registers (D form)	LD1 (multiple structures)	6	2	2*L, (I)	(decoder line break)
Load multiple 1-element structures to 3 registers (Q form)	LD1 (multiple structures)	6	1.33	3*L, (I)	decoder line break
Load multiple 1-element structures to 4 registers (D form)	LD1 (multiple structures)	6	2	2*L, (I)	(decoder line break)
Load multiple 1-element structures to 4 registers (Q form)	LD1 (multiple structures)	6	1	4*L, (I)	decoder line break

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load single 1-element structure (one lane)	LD1 (single structure)	8	4	L, V, (I)	(decoder line break)
Load single 1-element structure (all lanes)	LD1R	6	4	L, (I)	
Load multiple 2-element structures (D form)	LD2 (multiple structures)	8	3	L, 2*V, (I)	decoder line break
Load multiple 2-element structures (Q form)	LD2 (multiple structures)	8	2	2*L, 2*V, (I)	decoder line break
Load single 2-element structure (one lane)	LD2 (single structure)	8	3	L, 2*V, (I)	decoder line break
Load single 2-element structure (all lanes)	LD2R	8	3	L, 2*V, (I)	decoder line break
Load multiple 3-element structures (D form)	LD3 (multiple structures)	8	2	2*L, 3*V, (I)	decoder line break
Load multiple 3-element structures (Q form)	LD3 (multiple structures)	8	1.33	3*L, 3*V, (I)	decoder line break
Load single 3-element structure (one lane)	LD3 (single structure)	8	2	2*L, 3*V, (I)	decoder line break
Load single 3-element structure (all lanes)	LD3R	8	2	2*L, 3*V, (I)	decoder line break
Load multiple 4-element structures (D form)	LD4 (multiple structures)	8	1.5	2*L, 4*V, (I)	decoder line break
Load multiple 4-element structures (Q form)	LD4 (multiple structures)	9	0.75	4*L, 8*V, (I)	decoder line break
Load single 4-element structure (one lane, B/H/S)	LD4 (single structure)	8	1.5	L, 4*V, (I)	decoder line break
Load single 4-element structure (one lane, D)	LD4 (single structure)	8	1.5	2*L, 4*V, (I)	decoder line break
Load single 4-element structure (all lanes, B/H/S)	LD4R	8	1.5	L, 4*V, (I)	decoder line break
Load single 4-element structure (all lanes, D)	LD4R	8	1.5	2*L, 4*V, (I)	decoder line break

3.17 SIMD Store Instructions

AArch64 SIMD Store Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store multiple 1-element structures from 1 register	ST1 (multiple structures)	3	2	SA, V0123, (I)	
Store multiple 1-element structures from 2 registers (D form)	ST1 (multiple structures)	3	2	SA, V0123, (I)	
Store multiple 1-element structures from 2 registers (Q form)	ST1 (multiple structures)	3	1	SA, V0123, (I)	
Store multiple 1-element structures from 3 registers (D form)	ST1 (multiple structures)	4	1	2*SA, 2*V0123, (I)	(decoder line break)
Store multiple 1-element structures from 3 registers (Q form)	ST1 (multiple structures)	4	0.67	2*SA, 2*V0123, (I)	(decoder line break)
Store multiple 1-element structures from 4 registers (D form)	ST1 (multiple structures)	4	1	2*SA, 2*V0123, (I)	(decoder line break)
Store multiple 1-element structures from 4 registers (Q form)	ST1 (multiple structures)	4	0.5	2*SA, 2*V0123, (I)	(decoder line break)
Store single 1-element structure (one lane)	ST1 (single structure)	5	2	SA, V, V0123, (I)	(decoder line break)
Store multiple 2-element structures (D form)	ST2 (multiple structures)	5	2	SA, V, V0123, (I)	(decoder line break)
Store multiple 2-element structures (Q form)	ST2 (multiple structures)	5	1	SA, 2*V, V0123, (I)	decoder line break
Store single 2-element structure (one lane)	ST2 (single structure)	5	2	SA, V, V0123, (I)	(decoder line break)
Store multiple 3-element structures (D form)	ST3 (multiple structures)	5	1	2*SA, 2*V, 2*V0123, (I)	decoder line break
Store multiple 3-element structures (Q form)	ST3 (multiple structures)	6	0.67	2*SA, 3*V, 2*V0123, (I)	decoder line break
Store single 3-element structure (one lane)	ST3 (single structure)	5	1	2*SA, 2*V, 2*V0123, (I)	decoder line break
Store multiple 4-element structures (D form)	ST4 (multiple structures)	7	1	SA, 4*V, V0123, (I)	decoder line break
Store multiple 4-element structures (Q form, D element)	ST4 (multiple structures)	6	0.5	2*SA, 4*V, 2*V0123, (I)	decoder line break

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store multiple 4-element structures (Q form, B/H/S element)	ST4 (multiple structures)	8	0.5	2*SA, 8*V, 2*V0123, (I)	decoder line break
Store single 4-element structure (one lane, D)	ST4 (single structure)	5	1	SA, 2*V, V0123, (I)	decoder line break
Store single 4-element structure (one lane, B/H/S)	ST4 (single structure)	7	2	SA, 2*V, V0123, (I)	decoder line break

3.18 SVE Predicate Instructions

AArch64 SVE Predicate Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Logical	AND (predicates)	1	2	M	
	ANDS				
	BIC (predicates)				
	BICS				
	CTERMEQ, CTERMNE				
	EOR (predicates)				
	EORS				
	NAND				
	NANDS				
	NOR				
	NORS				
	ORN (predicates)				
	ORNS				
	ORR (predicates)				
	ORRS				
	PTEST				
SEL (predicates)					
	BRKA				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Control, counting	BRKAS	2	2	M	
	BRKB				
	BRKBS				
	BRKN				
	BRKNS				
	BRKPA				
	BRKPAS				
	BRKPB				
	BRKPBS				
	CNTB, CNTD, CNTH, CNTW				
	CNTP (predicate)				
	DECP (scalar)				
	SQDECP (scalar)				
	UQDECP (scalar)				
	INCP (scalar)				
	SQINCP (scalar)				
	UQINCP (scalar)				
	SQDECB				
	SQDECD (scalar)				
	SQDECH (scalar)				
	SQDECW (scalar)				
	UQDECB				
	UQDECD (scalar)				
	UQDECH (scalar)				
	UQDECW (scalar)				
	SQINCB				
SQINCD (scalar)					
SQINCH (scalar)					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SQINCW (scalar)				
	UQINCB				
	UQINCD (scalar)				
	UQINCH (scalar)				
	UQINCW (scalar)				
	PFIRST				
	PNEXT				
	WHILEGE (predicate)				
	WHILEGT (predicate)				
	WHILEHI (predicate)				
	WHILEHS (predicate)				
	WHILELE (predicate)				
	WHILELO (predicate)				
	WHILELS (predicate)				
	WHILELT (predicate)				
	WHILERW				
WHILEWR					
Misc processing	PFALSE	2	2	M	
	PTRUE (predicate)				
	PTRUES				
	PUNPKHI, PUNPKLO				
	REV (predicate)				
	TRN1, TRN2 (predicates)				
	UZP1, UZP2 (predicates)				
	ZIP1, ZIP2 (predicates)				
VL agnostic control	RDVL	1	8	I	
	ADDPL				
	ADDVL				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Predicate counting scalar (ALL #{1,2,4,8})	DECB, DECD, DECH, DECW (scalar)	1	8	I	
	INCB, INCD, INCH, INCW (scalar)				
Predicate counting scalar (other)	DECB, DECD, DECH, DECW (scalar)	2	2	M	
	INCB, INCD, INCH, INCW (scalar)				
Predicate counting vector	DECP (vector)	7	1	2*M, V	decoder line break
	SQDECP (vector)				
	UQDECP (vector)				
	INCP (vector)				
	SQINCP (vector)				
	UQINCP (vector)				
Read first fault reg (unpredicated)	RDFFR (unpredicated)	2	1	M0	
Read first fault reg (predicated)	RDFFR (predicated)	3	1	M, M0	
	RDFFRS				
Write/Set first fault reg	WRFFR	1	1	M0	
	SETFFR				

3.19 SVE Floating-Point Instructions

AArch64 SVE Floating-Point Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	FABD				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP Arithmetic	FABS	2	6	V	
	FADD (immediate)				
	FADD (vectors, predicated)				
	FADDP				
	FADD (vectors, unpredicated)				
	FAMAX				
	FMAX (immediate)				
	FMAX (vectors)				
	FMAXNM (immediate)				
	FMAXNM (vectors)				
	FMAXNMP				
	FMAXP				
	FAMIN				
	FMIN (immediate)				
	FMIN (vectors)				
	FMINNM (immediate)				
	FMINNM (vectors)				
	FMINNMP				
	FMINP				
	FCPY				
	FDUP				
	FNEG				
	FSUB (immediate)				
	FSUB (vectors, predicated)				
	FSUBR (immediate)				
	FSUBR (vectors)				
	FSUB (vectors, unpredicated)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP associative add, F16	FADDA	11	0.2	V03	multi-pumped
FP associative add, F32	FADDA	7	0.33	V03	multi-pumped
FP associative add, F64	FADDA	4	3	2*V	
FP compare	FAC<cc>	2	2	V03	
	FCM<cc> (zero)				
	FCM<cc> (vectors)				
FP multiply	FMUL (immediate)	3	6	V	SIMD multiply accumulate pipelines support late forwarding of the result from SIMD multiply μ Ops to the accumulate operand. The latency between the issue of the multiply producer and the multiply accumulate consumer is 2 cycles in this case.
	FMUL (vectors, predicated)				
	FMULX				
	FMUL (vectors, unpredicated)				
	FMUL (indexed)				
FP misc	FCADD	3	6	V	
	FSCALE				
	FTSMUL				
	FTSSEL				
FP multiply accumulate	FMLA (vectors)	4 (2)	6	V	
	FMLS (vectors)				
	FNMLA				
	FNMLS				
	FMLA (indexed)				
	FMLALB (vectors, FP16 to FP32)				
	FMLALB (indexed, FP16 to FP32)				
	FMLALT (vectors, FP16 to FP32)				
	FMLALT (indexed, FP16 to FP32)				
	FMLS (indexed)				
FMLSLB (vectors)					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	FMLSLB (indexed)				
	FMLSLT (vectors)				
	FMLSLT (indexed)				
	FMAD				
	FNMAD				
	FMSB				
	FNMSB				
	FRECPS				
	FRSQRTS				
	FTMAD				
FP complex multiply accumulate	FCMLA (vectors)	5 (2)	6	V	
	FCMLA (indexed)				
FP convert (to/from F64)	FCVT				
	FCVTLT	3	4	V0123	
	FCVTNT (predicated)				
	FCVTX				
	FCVTXNT				
FP convert (F32 to/from F16)	FCVT	4	2	V0123	multi-pumped
	FCVTLT				
	FCVTNT (predicated)				
FP logarithm, estimates, round, int convert (F16)	FCVTZS				
	FCVTZU				
	FLOGB	6	1	V0123	multi-pumped
	FRECPE				
	FRECPX				
	FRSQRTE				
	FRINT<r>				
	FCVTZS				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP logarithm, estimates, round, int convert (F32)	FCVTZU	4	2	V0123	multi-pumped
	FLOGB				
	FRECPE				
	FRECPX				
	FRSQRTE				
	FRINT<r>				
FP logarithm, estimates, round, int convert (F64)	FCVTZS	3	4	V0123	
	FCVTZU				
	FLOGB				
	FRECPE				
	FRECPX				
	FRSQRTE				
	FRINT<r>				
FP exponent	FEXPA	3	2	V12	
FP divide/sqrt (F16)	FDIV	13	0.25	V12	multi-pumped
	FDIVR				
	FSQRT				
FP divide/sqrt (F32)	FDIV	11	0.5	V12	multi-pumped
	FDIVR				
	FSQRT				
FP divide/sqrt (F64)	FDIV	14	1	V12	multi-pumped
	FDIVR				
	FSQRT				
FP reduction (F16)	FADDV	12	1.5	4*V	decoder line break
	FMAXNMV				
	FMAXV				
	FMINNMV				
	FMINV				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
FP reduction (F32)	FADDV	9	2	3*V	decoder line break
	FMAXNMV				
	FMAXV				
	FMINNMV				
	FMINV				
FP reduction (F64)	FADDV	6	3	2*V	
	FMAXNMV				
	FMAXV				
	FMINNMV				
	FMINV				

3.20 SVE FP8 Instructions

AArch64 SVE FP8 Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Convert (to/from F16)	F1CVT, F2CVT	4	2	V0123	multi-pumped
	F1CVTLT, F2CVTLT				
	FCVTN				
Convert (from F32)	FCVTNB	3	4	V0123	
	FCVTNT (unpredicated)				
Dot product, Multiply accumulate	FDOT (4-way, vectors)	4 (2)	6	V	
	FDOT (4-way, indexed)				
	FDOT (2-way, vectors, FP8 to FP16)				
	FDOT (2-way, indexed, FP8 to FP16)				
	FMLALB (vectors, FP8 to FP16)				
	FMLALB (indexed, FP8 to FP16)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	FMLALLBB (vectors)				
	FMLALLBB (indexed)				
	FMLALLBT (vectors)				
	FMLALLBT (indexed)				
	FMLALLTB (vectors)				
	FMLALLTB (indexed)				
	FMLALLTT (vectors)				
	FMLALLTT (indexed)				
	FMLALT (vectors, FP8 to FP16)				
	FMLALT (indexed, FP8 to FP16)				

3.21 SVE BF16 Instructions

AArch64 SVE BF16 Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Convert (to/from F8)	BF1CVT, BF2CVT	4	2	V0123	multi-pumped
	BF1CVTLT, BF2CVTLT				
	BFCVTN				
Convert (from F32)	BFCVT	4	4	V0123	
	BFCVTNT				
Multiply accumulate	BFMLALB (vectors)	4 (2)	6	V	
	BFMLALB (indexed)				
	BFMLALT (vectors)				
	BFMLALT (indexed)				
Dot product	BFDOT (vectors)	5 (3)	6	V	
	BFDOT (indexed)				
Matrix multiply accumulate	BFMMLA	6 (4)	6	V	

3.22 SVE Integer Instructions

AArch64 SVE Integer Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	ABS				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	ADCLB				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Misc	ADCLT	2	6	V	
	ADD (vectors, predicated)				
	ADDP				
	ADD (immediate)				
	ADD (vectors, unpredicated)				
	CADD				
	ADDHNB				
	ADDHNT				
	RADDHNB				
	RADDHNT				
	ADR				
	AND (vectors, predicated)				
	AND (immediate)				
	AND (vectors, unpredicated)				
	ASR (immediate, predicated)				
	ASR (wide element, predicated)				
	ASR (vectors)				
	ASRR				
	ASR (immediate, unpredicated)				
	ASR (wide element, unpredicated)				
	BIC (vectors, predicated)				
	BIC (vectors, unpredicated)				
	BSL1N				
	BSL2N				
	BSL				
	CLS				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	CLZ				
	CNOT				
	CNT				
	CPY (immediate, zeroing)				
	CPY (immediate, merging)				
	CPY (SIMD&FP scalar)				
	DECD, DECH, DECW (vector)				
	DUP (immediate)				
	DUP (indexed)				
	DUPM				
	EOR (vectors, predicated)				
	EOR (immediate)				
	EOR (vectors, unpredicated)				
	EORBT				
	EORTB				
	EXT				
	HISTCNT				
	HISTSEG				
	INCD, INCH, INCW (vector)				
	INSR (SIMD&FP scalar)				
	LSL (immediate, predicated)				
	LSL (wide element, predicated)				
	LSL (vectors)				
	LSLR				
	LSR (immediate, predicated)				
	LSR (wide element, predicated)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LSR (vectors)				
	LSRR				
	LSL (immediate, unpredicated)				
	LSL (wide elements, unpredicated)				
	LSR (immediate, unpredicated)				
	LSR (wide elements, unpredicated)				
	LUTI2				
	LUTI4				
	MOVPRFX (predicated)				
	MOVPRFX (unpredicated)				
	NBSL				
	NEG				
	NOT (vector)				
	ORR (vectors, predicated)				
	ORR (immediate)				
	ORR (vectors, unpredicated)				
	PMUL				
	PMULLB				
	PMULLT				
	RBIT				
	REV (vector)				
	REVB, REVH, REVW				
	SBCLB				
	SBCLT				
	SEL (vectors)				
	SHRNB				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SHRNT				
	SLI				
	SRI				
	SUB (vectors, predicated)				
	SUBR (vectors)				
	SUB (immediate)				
	SUB (vectors, unpredicated)				
	SUBR (immediate)				
	RSUBHNB				
	RSUBHNT				
	SUBHNB				
	SUBHNT				
	TBL				
	TBX				
	TRN1, TRN2 (vectors)				
	UZP1, UZP2 (vectors)				
	ZIP1, ZIP2 (vectors)				
	SABD				
	UABD				
	SABDLB				
	SABDLT				
	UABDLB				
	UABDLT				
	SADDLB				
	SADDLBT				
	SADDLT				
	SADDWB				
	SADDWT				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	UADDLB				
	UADDLT				
	UADDWB				
	UADDWT				
	SHADD				
	SRHADD				
	UHADD				
	URHADD				
	SHSUB				
	SHSUBR				
	UHSUB				
	UHSUBR				
	SMAX (vectors)				
	SMAXP				
	UMAX (vectors)				
	UMAXP				
	SMAX (immediate)				
	UMAX (immediate)				
	SMIN (vectors)				
	SMINP				
	UMIN (vectors)				
	UMINP				
	SMIN (immediate)				
	UMIN (immediate)				
	SQADD (vectors, predicated)				
	UQADD (vectors, predicated)				
	USQADD				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SQABS				
	SQADD (immediate)				
	SQADD (vectors, unpredicated)				
	SQCADD				
	UQADD (immediate)				
	UQADD (vectors, unpredicated)				
	SQDECD (vector)				
	SQDECH (vector)				
	SQDECW (vector)				
	UQDECD (vector)				
	UQDECH (vector)				
	UQDECW (vector)				
	SQINCD (vector)				
	SQINCH (vector)				
	SQINCW (vector)				
	UQINCD (vector)				
	UQINCH (vector)				
	UQINCW (vector)				
	SQNEG				
	SQSUB (vectors, predicated)				
	SQSUBR				
	UQSUB (vectors, predicated)				
	UQSUBR				
	SQSUB (immediate)				
	SQSUB (vectors, unpredicated)				
	UQSUB (immediate)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	UQSUB (vectors, unpredicated)				
	SSHLLB				
	SSHLLT				
	USHLLB				
	USHLLT				
	SSUBLB				
	SSUBLBT				
	SSUBLT				
	SSUBLTB				
	SSUBWB				
	SSUBWT				
	USUBLB				
	USUBLT				
	USUBWB				
	USUBWT				
	SUNPKHI, SUNPKLO				
	UUNPKHI, UUNPKLO				
	SUQADD				
	SXTB, SXTH, SXTW				
	UXTB, UXTH, UXTW				
	ASRD	s			

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Shift (complex)	RSHRNB		6	V	
	RSHRNT				
	SQRSHLR				
	SQRSHL				
	SQSHL (immediate)				
	SQSHL (vectors)				
	SQSHLR				
	SQSHLU				
	SRSHL				
	SRSHLR				
	UQRSHLR				
	UQRSHL				
	UQSHL (immediate)				
	UQSHL (vectors)				
	UQSHLR				
	URSHL				
	URSHLR				
	RSHRNB				
	RSHRNT				
	SQRSHRNB				
	SQRSHRNT				
	SQRSHRUNB				
	SQRSHRUNT				
	SQSHRNB				
	SQSHRNT				
	SQSHRUNB				
	SQSHRUNT				
	UQRSHRNB				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	UQRSHRNT				
	UQSHRNB				
	UQSHRNT				
	SRSHR				
	URSHR				
	SQXTNB				
	SQXTNT				
	SQXTUNB				
	SQXTUNT				
	UQXTNB				
	UQXTNT				
Shift and accumulate	SRSRA	4 (2)	6	V	
	SSRA				
	URSRA				
	USRA				
Multiply (B/H/S)	MUL (vectors, predicated)	4	4	V0123	
	SMULH (predicated)				
	UMULH (predicated)				
	MUL (immediate)				
	MUL (vectors, unpredicated)				
	SMULH (unpredicated)				
	SQDMULH (vectors)				
	SQRDMULH (vectors)				
	UMULH (unpredicated)				
	SQDMULH (indexed)				
	SQRDMULH (indexed)				
	MUL (indexed)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Multiply (D)	MUL (vectors, predicated)	5	2	V0123	multi-pumped
	SMULH (predicated)				
	UMULH (predicated)				
	MUL (vectors, unpredicated)				
	SMULH (unpredicated)				
	SQDMULH (vectors)				
	SQRDMULH (vectors)				
	UMULH (unpredicated)				
	MUL (immediate)				
	SQDMULH (indexed)				
	SQRDMULH (indexed)				
	MUL (indexed)				
	Multiply long				
SMULLB (indexed)					
SMULLT (vectors)					
SMULLT (indexed)					
SQDMULLB (vectors)					
SQDMULLB (indexed)					
SQDMULLT (vectors)					
SQDMULLT (indexed)					
UMULLB (vectors)					
UMULLB (indexed)					
UMULLT (vectors)					
UMULLT (indexed)					
		CMLA (vectors)			

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Multiply accumulate (B/H/S)	CMLA (indexed)	4 (2)	4	V0123	
	MLA (vectors)				
	MLS (vectors)				
	MLA (indexed)				
	MLS (indexed)				
	SQRDCMLAH (vectors)				
	SQRDMLAH (vectors)				
	SQRDMLSH (vectors)				
	SQRDCMLAH (indexed)				
	SQRDMLAH (indexed)				
	SQRDMLSH (indexed)				
Multiply accumulate (D)	CMLA (vectors)	5 (3)	2	V0123	multi-pumped
	MLA (vectors)				
	MLS (vectors)				
	MLA (indexed)				
	MLS (indexed)				
	SQRDCMLAH (vectors)				
	SQRDMLAH (vectors)				
	SQRDMLSH (vectors)				
	SQRDMLAH (indexed)				
	SQRDMLSH (indexed)				
	SMLALB (vectors)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Multiply accumulate long	SMLALB (indexed)	4 (2)	4	V0123	
	SMLALT (vectors)				
	SMLALT (indexed)				
	SMLS LB (vectors)				
	SMLS LB (indexed)				
	SMLS LT (vectors)				
	SMLS LT (indexed)				
	SQDMLALB (vectors)				
	SQDMLALB (indexed)				
	SQDMLALBT				
	SQDMLALT (vectors)				
	SQDMLALT (indexed)				
	SQDMLS LB (vectors)				
	SQDMLS LB (indexed)				
	SQDMLS LB T				
	SQDMLS LT (vectors)				
	SQDMLS LT (indexed)				
	UMLALB (vectors)				
	UMLALB (indexed)				
	UMLALT (vectors)				
UMLALT (indexed)					
UMLS LB (vectors)					
UMLS LB (indexed)					
UMLS LT (vectors)					
UMLS LT (indexed)					
Multiply add/sub	MAD	5 (3)	4	V0123	
	MSB				
	SABA				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Other arithmetic accumulate	SABALB	4 (2)	6	V	
	SABALT				
	UABA				
	UABALB				
	UABALT				
	SADALP				
	UADALP				
Dot product, Matrix multiply (8 bit)	CDOT (vectors)	3 (2)	6	V	
	CDOT (indexed)				
	SDOT (4-way, vectors)				
	UDOT (4-way, vectors)				
	SDOT (4-way, indexed)				
	UDOT (4-way, indexed)				
	USDOT (vectors)				
	SUDOT				
	USDOT (indexed)				
	SMMLA				
	UMMLA				
	USMMLA				
Dot product (16 bit)	CDOT (vectors)	3 (2)	4	V0123	
	CDOT (indexed)				
	SDOT (4-way, vectors)				
	UDOT (4-way, vectors)				
	SDOT (4-way, indexed)				
	UDOT (4-way, indexed)				
Bit manipulation	BDEP	6	1	2*V12	
	BEXT				
	BGRP				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Compare and set flags	CMP<cc> (immediate)	2	2	M, V03	The M pipeline is utilized only when the governing and destination predicates are the same.
	CMP<cc> (wide elements)				
	CMP<cc> (vectors)				
Extract (FP scalar and vector)	CLASTA (SIMD&FP scalar)	3	2	V12	
	CLASTA (vectors)				
	CLASTB (SIMD&FP scalar)				
	CLASTB (vectors)				
	LASTA (SIMD&FP scalar)				
	LASTB (SIMD&FP scalar)				
	COMPACT				
	SPLICE				
Extract (gen reg scalar conditional)	CLASTA (scalar)	8	2	M, V03, V12	decoder line break
	CLASTB (scalar)				
Extract (gen reg scalar unconditional)	LASTA (scalar)	6	2	V03, V12	
	LASTB (scalar)				
Convert int to FP (64b or to F64)	SCVTF	3	4	V0123	
	UCVTF				
Convert int to FP (32b to F16/F32)	SCVTF	4	2	V0123	multi-pumped
	UCVTF				
Convert int to FP (16b)	SCVTF	6	1	V0123	multi-pumped
	UCVTF				
Copy/Insert (from gen reg)	CPY (scalar)	5	2	M, V	
	INSR (scalar)				
Duplicate (from gen reg)	DUP (scalar)	3	2	M	
Divide	SDIV	7 (variable)	2	V45	SVE integer divides are performed using an iterative algorithm and block any subsequent similar operations to the same pipeline until complete. Latencies are up to 12 cycles for 32-bit

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	SDIVR				forms and up to 20 cycles for 64-bit forms.
	UDIV				
	UDIVR				
Index (immediates, B/H/S)	INDEX (immediates)	4	4	V0123	
Index (immediates, D)	INDEX (immediates)	5	2	V0123	multi-pumped
Index (scalar, B/H/S)	INDEX (immediate, scalar)	7	2	M, V0123	
	INDEX (scalar, immediate)				
	INDEX (scalars)				
Index (scalar, D)	INDEX (immediate, scalar)	8	2	M, V0123	multi-pumped
	INDEX (scalar, immediate)				
	INDEX (scalars)				
Matching operations	MATCH	2	2	M, V03	The M pipeline is utilized only when the governing and destination predicates are the same
	NMATCH				
Reciprocal estimate	URECPE	4	2	V0123	multi-pumped
	URSQRTE				
Reduction (logical)	ANDV	5	3	V, V0123	
	EORV				
	ORV				
Reduction (add, B form)	SADDV	10	1.5	2*V, 2*V0123	decoder line break
	UADDV				
Reduction (add, H form)	SADDV	8	2	V, 2*V0123	decoder line break
	UADDV				
Reduction (add, S form)	SADDV	7	2	2*V, V0123	decoder line break
	UADDV				
Reduction (min/max, B form)	SMAXV	8	2	V, 2*V0123	decoder line break
	UMAXV				
	SMINV				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	UMINV				
Reduction (min/max, H form)	SMAXV	7	2	2*V, V0123	decoder line break
	UMAXV				
	SMINV				
	UMINV				
Reduction (min/max, S form)	SMAXV	5	3	V, V0123	
	UMAXV				
	SMINV				
	UMINV				
Reduction (D form)	SMAXV	4	3	2*V	
	UMAXV				
	SMINV				
	UMINV				
	UADDV				

3.23 SVE Cryptography Instructions

AArch64 SVE Cryptography Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
AES, SHA3 operations	AESD	2	6	V	
	AESE				
	AESIMC				
	AESMC				
	BCAX				
	EOR3				
	RAX1				
	XAR				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
SM4 operations	SM4E	4	1	V0	
	SM4EKEY				

3.24 SVE Load Instructions

AArch64 SVE Load Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Load vector	LDR (vector)	6	4	L	
Load predicate	LDR (predicate)	5	2	M, L	
	LD1B (scalar plus immediate, single register)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Contiguous Load single structure	LD1D (scalar plus immediate, single register)	6	4	L	
	LD1H (scalar plus immediate, single register)				
	LD1RB				
	LD1RD				
	LD1RH				
	LD1RQB (scalar plus immediate)				
	LD1RQD (scalar plus immediate)				
	LD1RQH (scalar plus immediate)				
	LD1RQW (scalar plus immediate)				
	LD1RSB				
	LD1RSH				
	LD1RSW				
	LD1RW				
	LD1SB (scalar plus immediate)				
	LD1SH (scalar plus immediate)				
	LD1SW (scalar plus immediate)				
	LD1W (scalar plus immediate, single register)				
	LDNF1B				
	LDNF1D				
	LDNF1H				
	LDNF1SB				
LDNF1SH					
LDNF1SW					
LDNF1W					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LDNT1B (scalar plus immediate, single register)				
	LDNT1D (scalar plus immediate, single register)				
	LDNT1H (scalar plus immediate, single register)				
	LDNT1W (scalar plus immediate, single register)				
	LD1B (scalar plus scalar, single register)				
	LD1D (scalar plus scalar, single register)				
	LD1H (scalar plus scalar, single register)				
	LD1RQB (scalar plus scalar)				
	LD1RQD (scalar plus scalar)				
	LD1RQH (scalar plus scalar)				
	LD1RQW (scalar plus scalar)				
	LD1SB (scalar plus scalar)				
	LD1SH (scalar plus scalar)				
	LD1SW (scalar plus scalar)				
	LD1W (scalar plus scalar, single register)				
	LDFF1B (scalar plus scalar)				
	LDFF1D (scalar plus scalar)				
	LDFF1H (scalar plus scalar)				
	LDFF1SB (scalar plus scalar)				
	LDFF1SH (scalar plus scalar)				
	LDFF1SW (scalar plus scalar)				
	LDFF1W (scalar plus scalar)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LDNT1B (scalar plus scalar, single register)				
	LDNT1D (scalar plus scalar, single register)				
	LDNT1H (scalar plus scalar, single register)				
	LDNT1W (scalar plus scalar, single register)				
Contiguous Load 2 structures (scalar + immediate)	LD2B (scalar plus immediate)	8	2	2*L, 2*V	decoder line break
	LD2D (scalar plus immediate)				
	LD2H (scalar plus immediate)				
	LD2W (scalar plus immediate)				
Contiguous Load 2 structures (scalar + scalar)	LD2B (scalar plus scalar)	9	2	1, 2*L, 2*V	decoder line break
	LD2D (scalar plus scalar)				
	LD2H (scalar plus scalar)				
	LD2W (scalar plus scalar)				
Contiguous Load 3 structures (scalar + immediate)	LD3B (scalar plus immediate)	8	1.33	3*L, 3*V	decoder line break
	LD3D (scalar plus immediate)				
	LD3H (scalar plus immediate)				
	LD3W (scalar plus immediate)				
Contiguous Load 3 structures (scalar + scalar)	LD3B (scalar plus scalar)	9	1.33	1, 3*L, 3*V	decoder line break
	LD3D (scalar plus scalar)				
	LD3H (scalar plus scalar)				
	LD3W (scalar plus scalar)				
Contiguous Load 4 structures (scalar + immediate)	LD4B (scalar plus immediate)	9	0.75	4*L, 8*V	decoder line break
	LD4D (scalar plus immediate)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LD4H (scalar plus immediate)				
	LD4W (scalar plus immediate)				
Contiguous Load 4 structures (scalar + scalar)	LD4B (scalar plus scalar)	10	0.75	I, 4*L, 8*V	decoder line break
	LD4D (scalar plus scalar)				
	LD4H (scalar plus scalar)				
	LD4W (scalar plus scalar)				
Gather Load (vector + immediate, 32-bit element)	LD1B (vector plus immediate)	9	1	4*L, V03	decoder line break
	LD1H (vector plus immediate)				
	LD1W (vector plus immediate)				
	LDFF1B (vector plus immediate)				
	LDFF1H (vector plus immediate)				
	LDFF1W (vector plus immediate)				
	LD1SB (vector plus immediate)				
	LD1SH (vector plus immediate)				
	LDFF1SB (vector plus immediate)				
	LDFF1SH (vector plus immediate)				
Gather Load (vector + immediate, 64-bit element)	LD1B (vector plus immediate)	9	2	2*L, V03	decoder line break
	LD1H (vector plus immediate)				
	LD1W (vector plus immediate)				
	LDFF1B (vector plus immediate)				
	LDFF1H (vector plus immediate)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LDFF1W (vector plus immediate)				
	LD1D (vector plus immediate)				
	LDFF1D (vector plus immediate)				
	LD1SB (vector plus immediate)				
	LD1SH (vector plus immediate)				
	LDFF1SB (vector plus immediate)				
	LDFF1SH (vector plus immediate)				
	LD1SW (vector plus immediate)				
	LDFF1SW (vector plus immediate)				
Gather Load (scalar + vector, H form 32-bit scaled offset)	LD1H (scalar plus vector)	9	1	4*L, 2*V03	decoder line break
	LDFF1H (scalar plus vector)				
	LD1SH (scalar plus vector)				
	LDFF1SH (scalar plus vector)				
Gather Load (scalar + vector, other 32-bit element)	LD1B (scalar plus vector)	9	1	4*L, V03	decoder line break
	LD1H (scalar plus vector)				
	LD1W (scalar plus vector)				
	LDFF1B (scalar plus vector)				
	LDFF1H (scalar plus vector)				
	LDFF1W (scalar plus vector)				
	LD1SB (scalar plus vector)				
	LD1SH (scalar plus vector)				
	LDFF1SB (scalar plus vector)				
	LDFF1SH (scalar plus vector)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Gather Load (scalar + vector, 64-bit element)	LD1B (scalar plus vector)	9	2	2*L, V03	decoder line break
	LD1H (scalar plus vector)				
	LD1W (scalar plus vector)				
	LDFF1B (scalar plus vector)				
	LDFF1H (scalar plus vector)				
	LDFF1W (scalar plus vector)				
	LD1D (scalar plus vector)				
	LDFF1D (scalar plus vector)				
	LD1SB (scalar plus vector)				
	LD1SH (scalar plus vector)				
	LDFF1SB (scalar plus vector)				
	LDFF1SH (scalar plus vector)				
	LD1SW (scalar plus vector)				
	LDFF1SW (scalar plus vector)				
Gather non-temporal Load (32-bit element)	LDNT1B (vector plus scalar)	9	1	4*L, V03	decoder line break
	LDNT1H (vector plus scalar)				
	LDNT1W (vector plus scalar)				
	LDNT1SB				
	LDNT1SH				
Gather non-temporal Load (64-bit element)	LDNT1B (vector plus scalar)	9	2	2*L, V03	decoder line break
	LDNT1H (vector plus scalar)				
	LDNT1W (vector plus scalar)				
	LDNT1SB				
	LDNT1SH				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	LDNT1SW				
	LDNT1D (vector plus scalar)				

3.25 SVE Store Instructions

AArch64 SVE SVE Store Instructions

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Store from predicate	STR (predicate)	1	2	SA, D	
Store from vector	STR (vector)	3	2	SA, V0123	
Contiguous Store single structure	ST1B (scalar plus immediate, single register)	3	2	SA, V0123	
	ST1D (scalar plus immediate, single register)				
	ST1H (scalar plus immediate, single register)				
	ST1W (scalar plus immediate, single register)				
	STNT1B (scalar plus immediate, single register)				
	STNT1D (scalar plus immediate, single register)				
	STNT1H (scalar plus immediate, single register)				
	STNT1W (scalar plus immediate, single register)				
	ST1B (scalar plus scalar, single register)				
	ST1D (scalar plus scalar, single register)				
	ST1H (scalar plus scalar, single register)				
	ST1W (scalar plus scalar, single register)				
STNT1B (scalar plus scalar, single register)					

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
	STNT1D (scalar plus scalar, single register)				
	STNT1H (scalar plus scalar, single register)				
	STNT1W (scalar plus scalar, single register)				
Contiguous Store 2 structures (scalar + immediate)	ST2B (scalar plus immediate)	5	1	2*SA, 2*V, 2*V0123	decoder line break
	ST2D (scalar plus immediate)				
	ST2H (scalar plus immediate)				
	ST2W (scalar plus immediate)				
Contiguous Store 2 structures (scalar + scalar)	ST2B (scalar plus scalar)	5	1	1, 2*SA, 2*V, 2*V0123	decoder line break
	ST2D (scalar plus scalar)				
	ST2H (scalar plus scalar)				
	ST2W (scalar plus scalar)				
Contiguous Store 3 structures (scalar + immediate)	ST3B (scalar plus immediate)	6	0.67	3*SA, 3*V, 3*V0123	decoder line break
	ST3D (scalar plus immediate)				
	ST3H (scalar plus immediate)				
	ST3W (scalar plus immediate)				
Contiguous Store 3 structures (scalar + scalar)	ST3B (scalar plus scalar)	6	0.67	1, 3*SA, 3*V, 3*V0123	decoder line break
	ST3D (scalar plus scalar)				
	ST3H (scalar plus scalar)				
	ST3W (scalar plus scalar)				
Contiguous Store 4 structures (scalar + immediate, B/H/W)	ST4B (scalar plus immediate)	8	0.5	4*SA, 8*V, 4*V0123	decoder line break
	ST4H (scalar plus immediate)				
	ST4W (scalar plus immediate)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Contiguous Store 4 structures (scalar + immediate, D)	ST4D (scalar plus immediate)	6	0.5	4*SA, 4*V, 4*V0123	decoder line break
Contiguous Store 4 structures (scalar + scalar, B/H/W)	ST4B (scalar plus scalar)	8	0.5	I, 4*SA, 8*V, 4*V0123	decoder line break
	ST4H (scalar plus scalar)				
	ST4W (scalar plus scalar)				
Contiguous Store 4 structures (scalar + scalar, D)	ST4D (scalar plus scalar)	6	0.5	I, 4*SA, 4*V, 4*V0123	decoder line break
Scatter Store (vector + immediate, 32-bit element)	ST1B (vector plus immediate)	5	0.5	4*SA, V03, 4*V0123	decoder line break
	ST1H (vector plus immediate)				
	ST1W (vector plus immediate)				
Scatter Store (vector + immediate, 64-bit element)	ST1B (vector plus immediate)	4	1	2*SA, V03, 2*V0123	decoder line break
	ST1H (vector plus immediate)				
	ST1W (vector plus immediate)				
	ST1D (vector plus immediate)				
Scatter Store (scalar + vector, H form 32-bit scaled offset)	ST1H (scalar plus vector)	5	0.5	4*SA, 2*V03, 4*V0123	decoder line break
Scatter Store (scalar + vector, other 32-bit element)	ST1B (scalar plus vector)	5	0.5	4*SA, V03, 4*V0123	decoder line break
	ST1H (scalar plus vector)				
	ST1W (scalar plus vector)				
Scatter Store (scalar + vector, 64-bit element)	ST1B (scalar plus vector)	4	1	2*SA, V03, 2*V0123	decoder line break
	ST1H (scalar plus vector)				
	ST1W (scalar plus vector)				
	ST1D (scalar plus vector)				
Scatter non-temporal Store (32-bit element)	STNT1B (vector plus scalar)	5	0.5	4*SA, V03, 4*V0123	decoder line break
	STNT1H (vector plus scalar)				
	STNT1W (vector plus scalar)				

Instruction Group	Instruction	Exec Latency	Exec Throughput	Utilized Pipelines	Notes
Scatter non-temporal Store (64-bit element)	STNT1B (vector plus scalar)	4	1	2*SA, V03, 2*V0123	decoder line break
	STNT1H (vector plus scalar)				
	STNT1W (vector plus scalar)				
	STNT1D (vector plus scalar)				

4. Special Considerations

4.1 Dispatch Constraints

The "Utilized Pipelines" column of the tables in the previous chapter describes the breakdown of instructions into MOPs. Dispatch of MOPs from the in-order portion to the out-of-order portion of the microarchitecture includes several constraints. It is important to consider these constraints during code generation to maximize the effective dispatch bandwidth and subsequent execution bandwidth.

The dispatch stage can simultaneously process MOPs in a single cycle up to the following limitations based on the type(s) of pipelines used, and whether there are 2 threads are running in a core. Note that for some pipeline types, each MOP contributes towards the evaluation against multiple constraints.

In the event there are more MOPs available to be dispatched in a given cycle than can be supported, MOPs will be dispatched in oldest to youngest age-order to the extent allowed by these limits.

It is important to note that the rows in the following table are not mutually exclusive such that all relevant rows need to be examined together to reach the limit as listed in the table.

Pipeline(s) used	Limit when 1 thread is running	Limit when 2 threads are running
Any Pipeline Type as listed in the Pipeline Types Table (see NOTE)	10	5
M or MO	6	3
MO	3	3
Conditional branch MOPs with dynamically changing directions	3	3
L or SA	8	4
SA	4	2
Any Pipeline Type starting with "V" as listed in the Pipeline Types Table	6	5
V0123, V03, V12, or V0	4	4
V03 or V0	4	2
V12	4	2

Pipeline(s) used	Limit when 1 thread is running	Limit when 2 threads are running
V45	2	2
V0	2	2

NOTE:

The following types of MOps do not count towards the limits shown in the first row of the table.

- i. MOps using the D pipeline
- ii. MOps using the I pipeline in a Branch and Link instruction
- iii. Address modification MOps marked as (I) in the previous chapter, when those MOps are in the same instruction as a MOp in the L pipeline.
- iv. MOps using the V0123 pipeline, when those MOps are in the same instruction as a MOp in the SA pipeline.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality. NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Arm

Arm, AMBA, and Arm Powered are registered trademarks of Arm Limited. Cortex, MPCore, and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. "Arm" is used to represent Arm Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited.; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS, and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Copyright

© 2025 NVIDIA Corporation. All rights reserved.