Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM 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 ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at http://www.arm.com/company/policies/trademarks.

Copyright © 2019 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

AArch32 -- Base Instructions (alphabetic order)

ADC, ADCS (immediate): Add with Carry (immediate).

ADC, ADCS (register): Add with Carry (register).

ADC, ADCS (register-shifted register): Add with Carry (register-shifted register).

ADD (immediate, to PC): Add to PC: an alias of ADR.

ADD, ADDS (immediate): Add (immediate).

ADD, ADDS (register): Add (register).

ADD, ADDS (register-shifted register): Add (register-shifted register).

ADD, ADDS (SP plus immediate): Add to SP (immediate).

ADD, ADDS (SP plus register): Add to SP (register).

ADR: Form PC-relative address.

AND, ANDS (immediate): Bitwise AND (immediate).

AND, ANDS (register): Bitwise AND (register).

AND, ANDS (register-shifted register): Bitwise AND (register-shifted register).

ASR (immediate): Arithmetic Shift Right (immediate): an alias of MOV, MOVS (register).

ASR (register): Arithmetic Shift Right (register): an alias of MOV, MOVS (register-shifted register).

ASRS (immediate): Arithmetic Shift Right, setting flags (immediate): an alias of MOV, MOVS (register).

ASRS (register): Arithmetic Shift Right, setting flags (register): an alias of MOV, MOVS (register-shifted register).

B: Branch.

BFC: Bit Field Clear.

BFI: Bit Field Insert.

BIC, BICS (immediate): Bitwise Bit Clear (immediate).

BIC, BICS (register): Bitwise Bit Clear (register).

BIC, BICS (register-shifted register): Bitwise Bit Clear (register-shifted register).

BKPT: Breakpoint.

BL, BLX (immediate): Branch with Link and optional Exchange (immediate).

BLX (register): Branch with Link and Exchange (register).

BX: Branch and Exchange.

BXJ: Branch and Exchange, previously Branch and Exchange Jazelle.

CBNZ, CBZ: Compare and Branch on Nonzero or Zero.

CLREX: Clear-Exclusive.

CLZ: Count Leading Zeros.

CMN (immediate): Compare Negative (immediate).

CMN (register): Compare Negative (register).

CMN (register-shifted register): Compare Negative (register-shifted register).

CMP (immediate): Compare (immediate).

CMP (register): Compare (register).

CMP (register-shifted register): Compare (register-shifted register).

CPS, CPSID, CPSIE: Change PE State.

CRC32: CRC32.

CRC32C: CRC32C.

CSDB: Consumption of Speculative Data Barrier.

DBG: Debug hint.

DCPS1: Debug Change PE State to EL1.

DCPS2: Debug Change PE State to EL2.

DCPS3: Debug Change PE State to EL3.

DMB: Data Memory Barrier.

DSB: Data Synchronization Barrier.

EOR, EORS (immediate): Bitwise Exclusive OR (immediate).

EOR, EORS (register): Bitwise Exclusive OR (register).

EOR, EORS (register-shifted register): Bitwise Exclusive OR (register-shifted register).

ERET: Exception Return.

ESB: Error Synchronization Barrier.

HLT: Halting Breakpoint.

HVC: Hypervisor Call.

ISB: Instruction Synchronization Barrier.

IT: If-Then.

LDA: Load-Acquire Word.

LDAB: Load-Acquire Byte.

LDAEX: Load-Acquire Exclusive Word.

LDAEXB: Load-Acquire Exclusive Byte.

LDAEXD: Load-Acquire Exclusive Doubleword.

LDAEXH: Load-Acquire Exclusive Halfword.

LDAH: Load-Acquire Halfword.

LDC (immediate): Load data to System register (immediate).

LDC (literal): Load data to System register (literal).

LDM (exception return): Load Multiple (exception return).

LDM (User registers): Load Multiple (User registers).

LDM, LDMIA, LDMFD: Load Multiple (Increment After, Full Descending).

LDMDA, LDMFA: Load Multiple Decrement After (Full Ascending).

LDMDB, LDMEA: Load Multiple Decrement Before (Empty Ascending).

LDMIB, LDMED: Load Multiple Increment Before (Empty Descending).

LDR (immediate): Load Register (immediate).

LDR (literal): Load Register (literal).

LDR (register): Load Register (register).

LDRB (immediate): Load Register Byte (immediate).

LDRB (literal): Load Register Byte (literal).

LDRB (register): Load Register Byte (register).

LDRBT: Load Register Byte Unprivileged.

LDRD (immediate): Load Register Dual (immediate).

LDRD (literal): Load Register Dual (literal).

LDRD (register): Load Register Dual (register).

LDREX: Load Register Exclusive.

LDREXB: Load Register Exclusive Byte.

LDREXD: Load Register Exclusive Doubleword.

LDREXH: Load Register Exclusive Halfword.

LDRH (immediate): Load Register Halfword (immediate).

LDRH (literal): Load Register Halfword (literal).

LDRH (register): Load Register Halfword (register).

LDRHT: Load Register Halfword Unprivileged.

LDRSB (immediate): Load Register Signed Byte (immediate).

LDRSB (literal): Load Register Signed Byte (literal).

LDRSB (register): Load Register Signed Byte (register).

LDRSBT: Load Register Signed Byte Unprivileged.

LDRSH (immediate): Load Register Signed Halfword (immediate).

LDRSH (literal): Load Register Signed Halfword (literal).

LDRSH (register): Load Register Signed Halfword (register).

LDRSHT: Load Register Signed Halfword Unprivileged.

LDRT: Load Register Unprivileged.

LSL (immediate): Logical Shift Left (immediate): an alias of MOV, MOVS (register).

LSL (register): Logical Shift Left (register): an alias of MOV, MOVS (register-shifted register).

LSLS (immediate): Logical Shift Left, setting flags (immediate): an alias of MOV, MOVS (register).

LSLS (register): Logical Shift Left, setting flags (register): an alias of MOV, MOVS (register-shifted register).

LSR (immediate): Logical Shift Right (immediate): an alias of MOV, MOVS (register).

LSR (register): Logical Shift Right (register): an alias of MOV, MOVS (register-shifted register).

LSRS (immediate): Logical Shift Right, setting flags (immediate): an alias of MOV, MOVS (register).

LSRS (register): Logical Shift Right, setting flags (register): an alias of MOV, MOVS (register-shifted register).

MCR: Move to System register from general-purpose register or execute a System instruction.

MCRR: Move to System register from two general-purpose registers.

MLA, MLAS: Multiply Accumulate.

MLS: Multiply and Subtract.

MOV, MOVS (immediate): Move (immediate).

MOV, MOVS (register): Move (register).

MOV, MOVS (register-shifted register): Move (register-shifted register).

MOVT: Move Top.

MRC: Move to general-purpose register from System register.

MRRC: Move to two general-purpose registers from System register.

MRS: Move Special register to general-purpose register.

MRS (Banked register): Move Banked or Special register to general-purpose register.

MSR (Banked register): Move general-purpose register to Banked or Special register.

MSR (immediate): Move immediate value to Special register.

MSR (register): Move general-purpose register to Special register.

MUL, MULS: Multiply.

MVN, MVNS (immediate): Bitwise NOT (immediate).

MVN, MVNS (register): Bitwise NOT (register).

MVN, MVNS (register-shifted register): Bitwise NOT (register-shifted register).

NOP: No Operation.

ORN, ORNS (immediate): Bitwise OR NOT (immediate).

ORN, ORNS (register): Bitwise OR NOT (register).

ORR, ORRS (immediate): Bitwise OR (immediate).

ORR, ORRS (register): Bitwise OR (register).

ORR, ORRS (register-shifted register): Bitwise OR (register-shifted register).

PKHBT, PKHTB: Pack Halfword.

PLD (literal): Preload Data (literal).

PLD, PLDW (immediate): Preload Data (immediate).

PLD, PLDW (register): Preload Data (register).

PLI (immediate, literal): Preload Instruction (immediate, literal).

PLI (register): Preload Instruction (register).

POP: Pop Multiple Registers from Stack.

POP (multiple registers): Pop Multiple Registers from Stack: an alias of LDM, LDMIA, LDMFD.

POP (single register): Pop Single Register from Stack: an alias of LDR (immediate).

PSSBB: Physical Speculative Store Bypass Barrier.

PUSH: Push Multiple Registers to Stack.

PUSH (multiple registers): Push multiple registers to Stack: an alias of STMDB, STMFD.

PUSH (single register): Push Single Register to Stack: an alias of STR (immediate).

QADD: Saturating Add.

QADD16: Saturating Add 16.

QADD8: Saturating Add 8.

QASX: Saturating Add and Subtract with Exchange.

QDADD: Saturating Double and Add.

QDSUB: Saturating Double and Subtract.

QSAX: Saturating Subtract and Add with Exchange.

QSUB: Saturating Subtract.

QSUB16: Saturating Subtract 16.

QSUB8: Saturating Subtract 8.

RBIT: Reverse Bits.

REV: Byte-Reverse Word.

REV16: Byte-Reverse Packed Halfword.

REVSH: Byte-Reverse Signed Halfword.

RFE, RFEDA, RFEDB, RFEIA, RFEIB: Return From Exception.

ROR (immediate): Rotate Right (immediate): an alias of MOV, MOVS (register).

ROR (register): Rotate Right (register): an alias of MOV, MOVS (register-shifted register).

RORS (immediate): Rotate Right, setting flags (immediate): an alias of MOV, MOVS (register).

RORS (register): Rotate Right, setting flags (register): an alias of MOV, MOVS (register-shifted register).

RRX: Rotate Right with Extend: an alias of MOV, MOVS (register).

RRXS: Rotate Right with Extend, setting flags: an alias of MOV, MOVS (register).

RSB, RSBS (immediate): Reverse Subtract (immediate).

RSB, RSBS (register): Reverse Subtract (register).

RSB, RSBS (register-shifted register): Reverse Subtract (register-shifted register).

RSC, RSCS (immediate): Reverse Subtract with Carry (immediate).

RSC, RSCS (register): Reverse Subtract with Carry (register).

RSC, RSCS (register-shifted register): Reverse Subtract (register-shifted register).

SADD16: Signed Add 16.

SADD8: Signed Add 8.

SASX: Signed Add and Subtract with Exchange.

SB: Speculation Barrier.

SBC, SBCS (immediate): Subtract with Carry (immediate).

SBC, SBCS (register): Subtract with Carry (register).

SBC, SBCS (register-shifted register): Subtract with Carry (register-shifted register).

SBFX: Signed Bit Field Extract.

SDIV: Signed Divide.

SEL: Select Bytes.

SETEND: Set Endianness.

SETPAN: Set Privileged Access Never.

SEV: Send Event.

SEVL: Send Event Local.

SHADD16: Signed Halving Add 16.

SHADD8: Signed Halving Add 8.

SHASX: Signed Halving Add and Subtract with Exchange.

SHSAX: Signed Halving Subtract and Add with Exchange.

SHSUB16: Signed Halving Subtract 16.

SHSUB8: Signed Halving Subtract 8.

SMC: Secure Monitor Call.

SMLABB, SMLABT, SMLATB, SMLATT: Signed Multiply Accumulate (halfwords).

SMLAD, SMLADX: Signed Multiply Accumulate Dual.

SMLAL, SMLALS: Signed Multiply Accumulate Long.

SMLALBB, SMLALBT, SMLALTB, SMLALTT: Signed Multiply Accumulate Long (halfwords).

SMLALD, SMLALDX: Signed Multiply Accumulate Long Dual.

SMLAWB, SMLAWT: Signed Multiply Accumulate (word by halfword).

SMLSD, SMLSDX: Signed Multiply Subtract Dual.

SMLSLD, SMLSLDX: Signed Multiply Subtract Long Dual.

SMMLA, SMMLAR: Signed Most Significant Word Multiply Accumulate.

SMMLS, SMMLSR: Signed Most Significant Word Multiply Subtract.

SMMUL, SMMULR: Signed Most Significant Word Multiply.

SMUAD, SMUADX: Signed Dual Multiply Add.

SMULBB, SMULBT, SMULTB, SMULTT: Signed Multiply (halfwords).

SMULL, SMULLS: Signed Multiply Long.

SMULWB, SMULWT: Signed Multiply (word by halfword).

SMUSD, SMUSDX: Signed Multiply Subtract Dual.

SRS, SRSDA, SRSDB, SRSIA, SRSIB: Store Return State.

SSAT: Signed Saturate.

SSAT16: Signed Saturate 16.

SSAX: Signed Subtract and Add with Exchange.

SSBB: Speculative Store Bypass Barrier.

SSUB16: Signed Subtract 16.

SSUB8: Signed Subtract 8.

STC: Store data to System register.

STL: Store-Release Word.

STLB: Store-Release Byte.

STLEX: Store-Release Exclusive Word.

STLEXB: Store-Release Exclusive Byte.

STLEXD: Store-Release Exclusive Doubleword.

STLEXH: Store-Release Exclusive Halfword.

STLH: Store-Release Halfword.

STM (User registers): Store Multiple (User registers).

STM, STMIA, STMEA: Store Multiple (Increment After, Empty Ascending).

STMDA, STMED: Store Multiple Decrement After (Empty Descending).

STMDB, STMFD: Store Multiple Decrement Before (Full Descending).

STMIB, STMFA: Store Multiple Increment Before (Full Ascending).

STR (immediate): Store Register (immediate).

STR (register): Store Register (register).

STRB (immediate): Store Register Byte (immediate).

STRB (register): Store Register Byte (register).

STRBT: Store Register Byte Unprivileged.

STRD (immediate): Store Register Dual (immediate).

STRD (register): Store Register Dual (register).

STREX: Store Register Exclusive.

STREXB: Store Register Exclusive Byte.

STREXD: Store Register Exclusive Doubleword.

STREXH: Store Register Exclusive Halfword.

STRH (immediate): Store Register Halfword (immediate).

STRH (register): Store Register Halfword (register).

STRHT: Store Register Halfword Unprivileged.

STRT: Store Register Unprivileged.

SUB (immediate, from PC): Subtract from PC: an alias of ADR.

SUB, SUBS (immediate): Subtract (immediate).

SUB, SUBS (register): Subtract (register).

SUB, SUBS (register-shifted register): Subtract (register-shifted register).

SUB, SUBS (SP minus immediate): Subtract from SP (immediate).

SUB, SUBS (SP minus register): Subtract from SP (register).

SVC: Supervisor Call.

SXTAB: Signed Extend and Add Byte.

SXTAB16: Signed Extend and Add Byte 16.

SXTAH: Signed Extend and Add Halfword.

SXTB: Signed Extend Byte.

SXTB16: Signed Extend Byte 16.

SXTH: Signed Extend Halfword.

TBB, TBH: Table Branch Byte or Halfword.

TEQ (immediate): Test Equivalence (immediate).

TEQ (register): Test Equivalence (register).

TEQ (register-shifted register): Test Equivalence (register-shifted register).

TSB CSYNC: Trace Synchronization Barrier.

TST (immediate): Test (immediate).

TST (register): Test (register).

TST (register-shifted register): Test (register-shifted register).

UADD16: Unsigned Add 16.

UADD8: Unsigned Add 8.

UASX: Unsigned Add and Subtract with Exchange.

UBFX: Unsigned Bit Field Extract.

UDF: Permanently Undefined.

UDIV: Unsigned Divide.

UHADD16: Unsigned Halving Add 16.

UHADD8: Unsigned Halving Add 8.

UHASX: Unsigned Halving Add and Subtract with Exchange.

UHSAX: Unsigned Halving Subtract and Add with Exchange.

UHSUB16: Unsigned Halving Subtract 16.

UHSUB8: Unsigned Halving Subtract 8.

UMAAL: Unsigned Multiply Accumulate Accumulate Long.

UMLAL, UMLALS: Unsigned Multiply Accumulate Long.

UMULL, UMULLS: Unsigned Multiply Long.

UQADD16: Unsigned Saturating Add 16.

UQADD8: Unsigned Saturating Add 8.

UQASX: Unsigned Saturating Add and Subtract with Exchange.

UQSAX: Unsigned Saturating Subtract and Add with Exchange.

UQSUB16: Unsigned Saturating Subtract 16.

UQSUB8: Unsigned Saturating Subtract 8.

USAD8: Unsigned Sum of Absolute Differences.

USADA8: Unsigned Sum of Absolute Differences and Accumulate.

USAT: Unsigned Saturate.

USAT16: Unsigned Saturate 16.

USAX: Unsigned Subtract and Add with Exchange.

USUB16: Unsigned Subtract 16.

USUB8: Unsigned Subtract 8.

UXTAB: Unsigned Extend and Add Byte.

UXTAB16: Unsigned Extend and Add Byte 16.

UXTAH: Unsigned Extend and Add Halfword.

UXTB: Unsigned Extend Byte.

UXTB16: Unsigned Extend Byte 16.

UXTH: Unsigned Extend Halfword.

WFE: Wait For Event.

WFI: Wait For Interrupt.

YIELD: Yield hint.

(old)   **htmldiff from-**   **(new)**

DSB

Data Synchronization Barrier is a memory barrier that ensures the completion of memory accesses, see *Data Synchronization Barrier (DSB)*.

It has encodings from the following instruction sets: A32 (A1) and T32 (T1) .

A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	1	1	1	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	(0)	(0)	0	1	0	0	!= 0x00			

option

A1

```
DSB{<c>}{<q>} {<option>}
```

```
// No additional decoding required
```

T1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	1	1	(1)	(1)	(1)	(1)	1	0	(0)	0	(1)	(1)	(1)	(1)	0	1	0	0	!= 0x00			

option

T1

```
DSB{<c>}{<q>} {<option>}
```

```
// No additional decoding required
```

For more information about the CONSTRAINED UNPREDICTABLE behavior, see *Architectural Constraints on UNPREDICTABLE behaviors*.

Assembler Symbols

<c>   For encoding A1: see *Standard assembler syntax fields*. Must be AL or omitted.

For encoding T1: see *Standard assembler syntax fields*.

<q>   See *Standard assembler syntax fields*.

<option>   Specifies an optional limitation on the barrier operation. Values are:

SY
Full system is the required shareability domain, reads and writes are the required access types, both before and after the barrier instruction. Can be omitted. This option is referred to as the full system barrier. Encoded as option = 0b1111.

ST
Full system is the required shareability domain, writes are the required access type, both before and after the barrier instruction. SYST is a synonym for ST. Encoded as option = 0b1110.

LD
Full system is the required shareability domain, reads are the required access type before the barrier instruction, and reads and writes are the required access types after the barrier instruction. Encoded as option = 0b1101.

ISH
Inner Shareable is the required shareability domain, reads and writes are the required access types, both before and after the barrier instruction. Encoded as option = 0b1011.

ISHST

Inner Shareable is the required shareability domain, writes are the required access type, both before and after the barrier instruction. Encoded as option = 0b1010.

ISHLD

Inner Shareable is the required shareability domain, reads are the required access type before the barrier instruction, and reads and writes are the required access types after the barrier instruction. Encoded as option = 0b1001.

NSH

Non-shareable is the required shareability domain, reads and writes are the required access, both before and after the barrier instruction. Encoded as option = 0b0111.

NSHST

Non-shareable is the required shareability domain, writes are the required access type both before and after the barrier instruction. Encoded as option = 0b0110.

NSHLD

Non-shareable is the required shareability domain, reads are the required access type before the barrier instruction, and reads and writes are the required access types after the barrier instruction. Encoded as option = 0b0101.

OSH

Outer Shareable is the required shareability domain, reads and writes are the required access types, both before and after the barrier instruction. Encoded as option = 0b0011.

OSHST

Outer Shareable is the required shareability domain, writes are the required access type, both before and after the barrier instruction. Encoded as option = 0b0010.

OSHLD

Outer Shareable is the required shareability domain, reads are the required access type before the barrier instruction, and reads and writes are the required access types after the barrier instruction. Encoded as option = 0b0001.

For more information on whether an access is before or after a barrier instruction, see *Data Synchronization Barrier (DSB)*. All other encodings of option are reserved, other than the values 0b0000 and 0b0100. reserved. All unsupported and reserved options must execute as a full system DSB operation, but software must not rely on this behavior.
The value 0b0000 is used to encode SSBB and the value 0b0100 is used to encode PSSBB.

The instruction supports the following alternative <option> values, but Arm recommends that software does not use these alternative values:

- SH as an alias for ISH.
- SHST as an alias for ISHST.
- UN as an alias for NSH.
- UNST as an alias for NSHST.

Operation

```
if ConditionPassed() then
    EncodingSpecificOperations();
    case option of
        when '0001'  domain = MBReqDomain_OuterShareable;  types = MBReqTypes_Reads;
        when '0010'  domain = MBReqDomain_OuterShareable;  types = MBReqTypes_Writes;
        when '0011'  domain = MBReqDomain_OuterShareable;  types = MBReqTypes_All;
        when '0101'  domain = MBReqDomain_Nonshareable;    types = MBReqTypes_Reads;
        when '0110'  domain = MBReqDomain_Nonshareable;    types = MBReqTypes_Writes;
        when '0111'  domain = MBReqDomain_Nonshareable;    types = MBReqTypes_All;
        when '1001'  domain = MBReqDomain_InnerShareable;  types = MBReqTypes_Reads;
        when '1010'  domain = MBReqDomain_InnerShareable;  types = MBReqTypes_Writes;
        when '1011'  domain = MBReqDomain_InnerShareable;  types = MBReqTypes_All;
        when '1101'  domain = MBReqDomain_FullSystem;      types = MBReqTypes_Reads;
        when '1110'  domain = MBReqDomain_FullSystem;      types = MBReqTypes_Writes;
        otherwise
            if      option == '0000' then SEE "SSBB";
            elsif   option == '0100' then SEE "PSSBB";
            else    domain = MBReqDomain_FullSystem;       types = MBReqTypes_All;

    if PSTATE.EL IN {EL0, EL1} && EL2Enabled() then
        if HCR.BSU == '11' then
            domain = MBReqDomain_FullSystem;
        if HCR.BSU == '10' && domain != MBReqDomain_FullSystem then
            domain = MBReqDomain_OuterShareable;
        if HCR.BSU == '01' && domain == MBReqDomain_Nonshareable then
            domain = MBReqDomain_InnerShareable;

    DataSynchronizationBarrier(domain, types);
```

(old)	htmldiff from-	(new)

AArch32 -- SIMD&FP Instructions (alphabetic order)

AESD: AES single round decryption.

AESE: AES single round encryption.

AESIMC: AES inverse mix columns.

AESMC: AES mix columns.

FLDM*X (FLDMDBX, FLDMIAX): FLDM*X.

FSTMDBX, FSTMIAX: FSTMX.

SHA1C: SHA1 hash update (choose).

SHA1H: SHA1 fixed rotate.

SHA1M: SHA1 hash update (majority).

SHA1P: SHA1 hash update (parity).

SHA1SU0: SHA1 schedule update 0.

SHA1SU1: SHA1 schedule update 1.

SHA256H: SHA256 hash update part 1.

SHA256H2: SHA256 hash update part 2.

SHA256SU0: SHA256 schedule update 0.

SHA256SU1: SHA256 schedule update 1.

VABA: Vector Absolute Difference and Accumulate.

VABAL: Vector Absolute Difference and Accumulate Long.

VABD (floating-point): Vector Absolute Difference (floating-point).

VABD (integer): Vector Absolute Difference (integer).

VABDL (integer): Vector Absolute Difference Long (integer).

VABS: Vector Absolute.

VACGE: Vector Absolute Compare Greater Than or Equal.

VACGT: Vector Absolute Compare Greater Than.

VACLE: Vector Absolute Compare Less Than or Equal: an alias of VACGE.

VACLT: Vector Absolute Compare Less Than: an alias of VACGT.

VADD (floating-point): Vector Add (floating-point).

VADD (integer): Vector Add (integer).

VADDHN: Vector Add and Narrow, returning High Half.

VADDL: Vector Add Long.

VADDW: Vector Add Wide.

VAND (immediate): Vector Bitwise AND (immediate): an alias of VBIC (immediate).

VAND (register): Vector Bitwise AND (register).

VBIC (immediate): Vector Bitwise Bit Clear (immediate).

VBIC (register): Vector Bitwise Bit Clear (register).

VBIF: Vector Bitwise Insert if False.

VBIT: Vector Bitwise Insert if True.

VBSL: Vector Bitwise Select.

VCADD: Vector Complex Add.

VCEQ (immediate #0): Vector Compare Equal to Zero.

VCEQ (register): Vector Compare Equal.

VCGE (immediate #0): Vector Compare Greater Than or Equal to Zero.

VCGE (register): Vector Compare Greater Than or Equal.

VCGT (immediate #0): Vector Compare Greater Than Zero.

VCGT (register): Vector Compare Greater Than.

VCLE (immediate #0): Vector Compare Less Than or Equal to Zero.

VCLE (register): Vector Compare Less Than or Equal: an alias of VCGE (register).

VCLS: Vector Count Leading Sign Bits.

VCLT (immediate #0): Vector Compare Less Than Zero.

VCLT (register): Vector Compare Less Than: an alias of VCGT (register).

VCLZ: Vector Count Leading Zeros.

VCMLA: Vector Complex Multiply Accumulate.

VCMLA (by element): Vector Complex Multiply Accumulate (by element).

VCMP: Vector Compare.

VCMPE: Vector Compare, raising Invalid Operation on NaN.

VCNT: Vector Count Set Bits.

VCVT (between double-precision and single-precision): Convert between double-precision and single-precision.

VCVT (between floating-point and fixed-point, Advanced SIMD): Vector Convert between floating-point and fixed-point.

VCVT (between floating-point and fixed-point, floating-point): Convert between floating-point and fixed-point.

VCVT (between floating-point and integer, Advanced SIMD): Vector Convert between floating-point and integer.

VCVT (between half-precision and single-precision, Advanced SIMD): Vector Convert between half-precision and single-precision.

VCVT (floating-point to integer, floating-point): Convert floating-point to integer with Round towards Zero.

VCVT (from single-precision to BFloat16, Advanced SIMD): Vector Convert from single-precision to BFloat16.

VCVT (integer to floating-point, floating-point): Convert integer to floating-point.

VCVTA (Advanced SIMD): Vector Convert floating-point to integer with Round to Nearest with Ties to Away.

VCVTA (floating-point): Convert floating-point to integer with Round to Nearest with Ties to Away.

VCVTB: Convert to or from a half-precision value in the bottom half of a single-precision register.

VCVTB (BFloat16): Converts from a single-precision value to a BFloat16 value in the bottom half of a single-precision register.

VCVTM (Advanced SIMD): Vector Convert floating-point to integer with Round towards -Infinity.

VCVTM (floating-point): Convert floating-point to integer with Round towards -Infinity.

VCVTN (Advanced SIMD): Vector Convert floating-point to integer with Round to Nearest.

VCVTN (floating-point): Convert floating-point to integer with Round to Nearest.

VCVTP (Advanced SIMD): Vector Convert floating-point to integer with Round towards +Infinity.

VCVTP (floating-point): Convert floating-point to integer with Round towards +Infinity.

VCVTR: Convert floating-point to integer.

VCVTT: Convert to or from a half-precision value in the top half of a single-precision register.

VCVTT (BFloat16): Converts from a single-precision value to a BFloat16 value in the top half of a single-precision register..

VDIV: Divide.

VDOT (by element): BFloat16 floating-point indexed dot product (vector, by element).

VDOT (vector): BFloat16 floating-point (BF16) dot product (vector).

VDUP (general-purpose register): Duplicate general-purpose register to vector.

VDUP (scalar): Duplicate vector element to vector.

VEOR: Vector Bitwise Exclusive OR.

VEXT (byte elements): Vector Extract.

VEXT (multibyte elements): Vector Extract: an alias of VEXT (byte elements).

VFMA: Vector Fused Multiply Accumulate.

VFMAB, VFMAT (BFloat16, by scalar): BFloat16 floating-point widening multiply-add long (by scalar).

VFMAB, VFMAT (BFloat16, vector): BFloat16 floating-point widening multiply-add long (vector).

VFMAL (by scalar): Vector Floating-point Multiply-Add Long to accumulator (by scalar).

VFMAL (vector): Vector Floating-point Multiply-Add Long to accumulator (vector).

VFMS: Vector Fused Multiply Subtract.

VFMSL (by scalar): Vector Floating-point Multiply-Subtract Long from accumulator (by scalar).

VFMSL (vector): Vector Floating-point Multiply-Subtract Long from accumulator (vector).

VFNMA: Vector Fused Negate Multiply Accumulate.

VFNMS: Vector Fused Negate Multiply Subtract.

VHADD: Vector Halving Add.

VHSUB: Vector Halving Subtract.

VINS: Vector move Insertion.

VJCVT: Javascript Convert to signed fixed-point, rounding toward Zero.

VLD1 (multiple single elements): Load multiple single 1-element structures to one, two, three, or four registers.

VLD1 (single element to all lanes): Load single 1-element structure and replicate to all lanes of one register.

VLD1 (single element to one lane): Load single 1-element structure to one lane of one register.

VLD2 (multiple 2-element structures): Load multiple 2-element structures to two or four registers.

VLD2 (single 2-element structure to all lanes): Load single 2-element structure and replicate to all lanes of two registers.

VLD2 (single 2-element structure to one lane): Load single 2-element structure to one lane of two registers.

VLD3 (multiple 3-element structures): Load multiple 3-element structures to three registers.

VLD3 (single 3-element structure to all lanes): Load single 3-element structure and replicate to all lanes of three registers.

VLD3 (single 3-element structure to one lane): Load single 3-element structure to one lane of three registers.

VLD4 (multiple 4-element structures): Load multiple 4-element structures to four registers.

VLD4 (single 4-element structure to all lanes): Load single 4-element structure and replicate to all lanes of four registers.

VLD4 (single 4-element structure to one lane): Load single 4-element structure to one lane of four registers.

VLDM, VLDMDB, VLDMIA: Load Multiple SIMD&FP registers.

VLDR (immediate): Load SIMD&FP register (immediate).

VLDR (literal): Load SIMD&FP register (literal).

VMAX (floating-point): Vector Maximum (floating-point).

VMAX (integer): Vector Maximum (integer).

VMAXNM: Floating-point Maximum Number.

VMIN (floating-point): Vector Minimum (floating-point).

VMIN (integer): Vector Minimum (integer).

VMINNM: Floating-point Minimum Number.

VMLA (by scalar): Vector Multiply Accumulate (by scalar).

VMLA (floating-point): Vector Multiply Accumulate (floating-point).

VMLA (integer): Vector Multiply Accumulate (integer).

VMLAL (by scalar): Vector Multiply Accumulate Long (by scalar).

VMLAL (integer): Vector Multiply Accumulate Long (integer).

VMLS (by scalar): Vector Multiply Subtract (by scalar).

VMLS (floating-point): Vector Multiply Subtract (floating-point).

VMLS (integer): Vector Multiply Subtract (integer).

VMLSL (by scalar): Vector Multiply Subtract Long (by scalar).

VMLSL (integer): Vector Multiply Subtract Long (integer).

VMMLA: BFloat16 floating-point matrix multiply-accumulate.

VMOV (between general-purpose register and half-precision): Copy 16 bits of a general-purpose register to or from a 32-bit SIMD&FP register.

VMOV (between general-purpose register and single-precision): Copy a general-purpose register to or from a 32-bit SIMD&FP register.

VMOV (between two general-purpose registers and a doubleword floating-point register): Copy two general-purpose registers to or from a SIMD&FP register.

VMOV (between two general-purpose registers and two single-precision registers): Copy two general-purpose registers to a pair of 32-bit SIMD&FP registers.

VMOV (general-purpose register to scalar): Copy a general-purpose register to a vector element.

VMOV (immediate): Copy immediate value to a SIMD&FP register.

VMOV (register): Copy between FP registers.

VMOV (register, SIMD): Copy between SIMD registers: an alias of VORR (register).

VMOV (scalar to general-purpose register): Copy a vector element to a general-purpose register with sign or zero extension.

VMOVL: Vector Move Long.

VMOVN: Vector Move and Narrow.

VMOVX: Vector Move extraction.

VMRS: Move SIMD&FP Special register to general-purpose register.

VMSR: Move general-purpose register to SIMD&FP Special register.

VMUL (by scalar): Vector Multiply (by scalar).

VMUL (floating-point): Vector Multiply (floating-point).

VMUL (integer and polynomial): Vector Multiply (integer and polynomial).

VMULL (by scalar): Vector Multiply Long (by scalar).

VMULL (integer and polynomial): Vector Multiply Long (integer and polynomial).

VMVN (immediate): Vector Bitwise NOT (immediate).

VMVN (register): Vector Bitwise NOT (register).

VNEG: Vector Negate.

VNMLA: Vector Negate Multiply Accumulate.

VNMLS: Vector Negate Multiply Subtract.

VNMUL: Vector Negate Multiply.

VORN (immediate): Vector Bitwise OR NOT (immediate): an alias of VORR (immediate).

VORN (register): Vector bitwise OR NOT (register).

VORR (immediate): Vector Bitwise OR (immediate).

VORR (register): Vector bitwise OR (register).

VPADAL: Vector Pairwise Add and Accumulate Long.

VPADD (floating-point): Vector Pairwise Add (floating-point).

VPADD (integer): Vector Pairwise Add (integer).

VPADDL: Vector Pairwise Add Long.

VPMAX (floating-point): Vector Pairwise Maximum (floating-point).

VPMAX (integer): Vector Pairwise Maximum (integer).

VPMIN (floating-point): Vector Pairwise Minimum (floating-point).

VPMIN (integer): Vector Pairwise Minimum (integer).

VPOP: Pop SIMD&FP registers from Stack: an alias of VLDM, VLDMDB, VLDMIA.

VPUSH: Push SIMD&FP registers to Stack: an alias of VSTM, VSTMDB, VSTMIA.

VQABS: Vector Saturating Absolute.

VQADD: Vector Saturating Add.

VQDMLAL: Vector Saturating Doubling Multiply Accumulate Long.

VQDMLSL: Vector Saturating Doubling Multiply Subtract Long.

VQDMULH: Vector Saturating Doubling Multiply Returning High Half.

VQDMULL: Vector Saturating Doubling Multiply Long.

VQMOVN, VQMOVUN: Vector Saturating Move and Narrow.

VQNEG: Vector Saturating Negate.

VQRDMLAH: Vector Saturating Rounding Doubling Multiply Accumulate Returning High Half.

VQRDMLSH: Vector Saturating Rounding Doubling Multiply Subtract Returning High Half.

VQRDMULH: Vector Saturating Rounding Doubling Multiply Returning High Half.

VQRSHL: Vector Saturating Rounding Shift Left.

VQRSHRN (zero): Vector Saturating Rounding Shift Right, Narrow: an alias of VQMOVN, VQMOVUN.

VQRSHRN, VQRSHRUN: Vector Saturating Rounding Shift Right, Narrow.

VQRSHRUN (zero): Vector Saturating Rounding Shift Right, Narrow: an alias of VQMOVN, VQMOVUN.

VQSHL (register): Vector Saturating Shift Left (register).

VQSHL, VQSHLU (immediate): Vector Saturating Shift Left (immediate).

VQSHRN (zero): Vector Saturating Shift Right, Narrow: an alias of VQMOVN, VQMOVUN.

VQSHRN, VQSHRUN: Vector Saturating Shift Right, Narrow.

VQSHRUN (zero): Vector Saturating Shift Right, Narrow: an alias of VQMOVN, VQMOVUN.

VQSUB: Vector Saturating Subtract.

VRADDHN: Vector Rounding Add and Narrow, returning High Half.

VRECPE: Vector Reciprocal Estimate.

VRECPS: Vector Reciprocal Step.

VREV16: Vector Reverse in halfwords.

VREV32: Vector Reverse in words.

VREV64: Vector Reverse in doublewords.

VRHADD: Vector Rounding Halving Add.

VRINTA (Advanced SIMD): Vector Round floating-point to integer towards Nearest with Ties to Away.

VRINTA (floating-point): Round floating-point to integer to Nearest with Ties to Away.

VRINTM (Advanced SIMD): Vector Round floating-point to integer towards -Infinity.

VRINTM (floating-point): Round floating-point to integer towards -Infinity.

VRINTN (Advanced SIMD): Vector Round floating-point to integer to Nearest.

VRINTN (floating-point): Round floating-point to integer to Nearest.

VRINTP (Advanced SIMD): Vector Round floating-point to integer towards +Infinity.

VRINTP (floating-point): Round floating-point to integer towards +Infinity.

VRINTR: Round floating-point to integer.

VRINTX (Advanced SIMD): Vector round floating-point to integer inexact.

VRINTX (floating-point): Round floating-point to integer inexact.

VRINTZ (Advanced SIMD): Vector round floating-point to integer towards Zero.

VRINTZ (floating-point): Round floating-point to integer towards Zero.

VRSHL: Vector Rounding Shift Left.

VRSHR: Vector Rounding Shift Right.

VRSHR (zero): Vector Rounding Shift Right: an alias of VORR (register).

VRSHRN: Vector Rounding Shift Right and Narrow.

VRSHRN (zero): Vector Rounding Shift Right and Narrow: an alias of VMOVN.

VRSQRTE: Vector Reciprocal Square Root Estimate.

VRSQRTS: Vector Reciprocal Square Root Step.

VRSRA: Vector Rounding Shift Right and Accumulate.

VRSUBHN: Vector Rounding Subtract and Narrow, returning High Half.

VSDOT (by element): Dot Product index form with signed integers..

VSDOT (vector): Dot Product vector form with signed integers..

VSELEQ, VSELGE, VSELGT, VSELVS: Floating-point conditional select.

VSHL (immediate): Vector Shift Left (immediate).

VSHL (register): Vector Shift Left (register).

VSHLL: Vector Shift Left Long.

VSHR: Vector Shift Right.

VSHR (zero): Vector Shift Right: an alias of VORR (register).

VSHRN: Vector Shift Right Narrow.

VSHRN (zero): Vector Shift Right Narrow: an alias of VMOVN.

VSLI: Vector Shift Left and Insert.

VSMMLA: Widening 8-bit signed integer matrix multiply-accumulate into 2x2 matrix.

VSQRT: Square Root.

VSRA: Vector Shift Right and Accumulate.

VSRI: Vector Shift Right and Insert.

VST1 (multiple single elements): Store multiple single elements from one, two, three, or four registers.

VST1 (single element from one lane): Store single element from one lane of one register.

VST2 (multiple 2-element structures): Store multiple 2-element structures from two or four registers.

VST2 (single 2-element structure from one lane): Store single 2-element structure from one lane of two registers.

VST3 (multiple 3-element structures): Store multiple 3-element structures from three registers.

VST3 (single 3-element structure from one lane): Store single 3-element structure from one lane of three registers.

VST4 (multiple 4-element structures): Store multiple 4-element structures from four registers.

VST4 (single 4-element structure from one lane): Store single 4-element structure from one lane of four registers.

VST4 (single 4-element structure from one lane): Store single 4-element structure from one lane of four registers.

VSTM, VSTMDB, VSTMIA: Store multiple SIMD&FP registers.

VSTR: Store SIMD&FP register.

VSUB (floating-point): Vector Subtract (floating-point).

VSUB (integer): Vector Subtract (integer).

VSUBHN: Vector Subtract and Narrow, returning High Half.

VSUBL: Vector Subtract Long.

VSUBW: Vector Subtract Wide.

VSUDOT (by element): Dot Product index form with signed and unsigned integers (by element).

VSWP: Vector Swap.

VTBL, VTBX: Vector Table Lookup and Extension.

VTRN: Vector Transpose.

VTST: Vector Test Bits.

VUDOT (by element): Dot Product index form with unsigned integers..

VUDOT (vector): Dot Product vector form with unsigned integers..

VUMMLA: Widening 8-bit unsigned integer matrix multiply-accumulate into 2x2 matrix.

VUSDOT (by element): Dot Product index form with unsigned and signed integers (by element).

VUSDOT (vector): Dot Product vector form with mixed-sign integers.

VUSMMLA: Widening 8-bit mixed integer matrix multiply-accumulate into 2x2 matrix.

VUZP: Vector Unzip.

VUZP (alias): Vector Unzip: an alias of VTRN.

VZIP: Vector Zip.

VZIP (alias): Vector Zip: an alias of VTRN.

(old)   **htmldiff from-**   **(new)**

VCVTA (Advanced SIMD)

Vector Convert floating-point to integer with Round to Nearest with Ties to Away converts each element in a vector from floating-point to integer using the Round to Nearest with Ties to Away rounding mode, and places the results in a second vector.

The operand vector elements are floating-point numbers.

The result vector elements are integers, 32-bit and the same size as the operand vector elements. integers. Signed and unsigned integers are distinct.

Depending on settings in the *CPACR*, *NSACR*, *HCPTR*, and *FPEXC* registers, and the Security state and PE mode in which the instruction is executed, an attempt to execute the instruction might be UNDEFINED, or trapped to Hyp mode. For more information see *Enabling Advanced SIMD and floating-point support*.

It has encodings from the following instruction sets: A32 (A1) and T32 (T1) .

A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	D	1	1	size		1	1	Vd				0	0	0	0	op	Q	M	0	Vm			

RM

64-bit SIMD vector (Q == 0)

VCVTA{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTA{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

T1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	D	1	1	size		1	1	Vd				0	0	0	0	op	Q	M	0	Vm		

RM

64-bit SIMD vector (Q == 0)

VCVTA{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTA{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if InITBlock() then UNPREDICTABLE;
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

CONSTRAINED UNPREDICTABLE behavior

If `InITBlock()`, then one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction executes as if it passes the Condition code check.
- The instruction executes as NOP. This means it behaves as if it fails the Condition code check.

Assembler Symbols

<q>   See *Standard assembler syntax fields*.

<dt>   Is the data type for the elements of the destination, encoded in "op:size":

op	size	<dt>
0	01S32	S16
01	10U32	S32
1	01	U16
1	10	U32

<dt2>   Is the data type for the elements of the source vector, encoded in "size":

size	<dt2>
01	F16
10	F32

<Qd>   Is the 128-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field as <Qd>*2.

<Qm>   Is the 128-bit name of the SIMD&FP source register, encoded in the "M:Vm" field as <Qm>*2.

<Dd>   Is the 64-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field.

<Dm>   Is the 64-bit name of the SIMD&FP source register, encoded in the "M:Vm" field.

Operation

```
EncodingSpecificOperations(); CheckAdvSIMDEnabled();
bits(esize) result;
for r = 0 to regs-1
    for e = 0 to elements-1
        Elem[D[d+r],e,esize] = FPToFixed(Elem[D[m+r],e,esize], 0, unsigned,
                                          StandardFPSCRValue(), rounding);
```

(old)	**htmldiff from-**	**(new)**

VCVTM (Advanced SIMD)

Vector Convert floating-point to integer with Round towards -Infinity converts each element in a vector from floating-point to integer using the Round towards -Infinity rounding mode, and places the results in a second vector.

The operand vector elements are floating-point numbers.

The result vector elements are integers, 32-bit and the same size as the operand vector elements. integers. Signed and unsigned integers are distinct.

Depending on settings in the *CPACR*, *NSACR*, *HCPTR*, and *FPEXC* registers, and the Security state and PE mode in which the instruction is executed, an attempt to execute the instruction might be UNDEFINED, or trapped to Hyp mode. For more information see *Enabling Advanced SIMD and floating-point support*.

It has encodings from the following instruction sets: A32 (A1) and T32 (T1) .

A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	D	1	1	size		1	1	Vd				0	0	1	1	op	Q	M	0	Vm			

RM

64-bit SIMD vector (Q == 0)

VCVTM{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTM{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

T1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	D	1	1	size		1	1	Vd				0	0	1	1	op	Q	M	0	Vm		

RM

64-bit SIMD vector (Q == 0)

VCVTM{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTM{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if InITBlock() then UNPREDICTABLE;
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

CONSTRAINED UNPREDICTABLE behavior

If `InITBlock()`, then one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction executes as if it passes the Condition code check.
- The instruction executes as NOP. This means it behaves as if it fails the Condition code check.

Assembler Symbols

<q>   See *Standard assembler syntax fields*.

<dt>   Is the data type for the elements of the destination, encoded in "op:size":

op	size	<dt>
0	01S32	S16
01	10U32	S32
1	01	U16
1	10	U32

<dt2>   Is the data type for the elements of the source vector, encoded in "size":

size	<dt2>
01	F16
10	F32

<Qd>   Is the 128-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field as <Qd>*2.

<Qm>   Is the 128-bit name of the SIMD&FP source register, encoded in the "M:Vm" field as <Qm>*2.

<Dd>   Is the 64-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field.

<Dm>   Is the 64-bit name of the SIMD&FP source register, encoded in the "M:Vm" field.

Operation

```
EncodingSpecificOperations(); CheckAdvSIMDEnabled();
bits(esize) result;
for r = 0 to regs-1
    for e = 0 to elements-1
        Elem[D[d+r],e,esize] = FPToFixed(Elem[D[m+r],e,esize], 0, unsigned,
                                         StandardFPSCRValue(), rounding);
```

(old)   **htmldiff from-**   **(new)**

VCVTN (Advanced SIMD)

Vector Convert floating-point to integer with Round to Nearest converts each element in a vector from floating-point to integer using the Round to Nearest rounding mode, and places the results in a second vector.

The operand vector elements are floating-point numbers.

The result vector elements are integers, 32-bit and the same size as the operand vector elements. integers. Signed and unsigned integers are distinct.

Depending on settings in the *CPACR*, *NSACR*, *HCPTR*, and *FPEXC* registers, and the Security state and PE mode in which the instruction is executed, an attempt to execute the instruction might be UNDEFINED, or trapped to Hyp mode. For more information see *Enabling Advanced SIMD and floating-point support*.

It has encodings from the following instruction sets: A32 (A1) and T32 (T1) .

A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	D	1	1	size		1	1	Vd				0	0	0	1	op	Q	M	0	Vm			

RM

64-bit SIMD vector (Q == 0)

VCVTN{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTN{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

T1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	D	1	1	size		1	1	Vd				0	0	0	1	op	Q	M	0	Vm		

RM

64-bit SIMD vector (Q == 0)

VCVTN{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTN{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if InITBlock() then UNPREDICTABLE;
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

CONSTRAINED UNPREDICTABLE behavior

If `InITBlock()`, then one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction executes as if it passes the Condition code check.
- The instruction executes as NOP. This means it behaves as if it fails the Condition code check.

Assembler Symbols

<q>   See *Standard assembler syntax fields*.

<dt>   Is the data type for the elements of the destination, encoded in "op:size":

op	size	<dt>
0	01 ~~S32~~	S16
0~~1~~	10 ~~U32~~	S32
1	01	U16
1	10	U32

<dt2>   Is the data type for the elements of the source vector, encoded in "size":

size	<dt2>
01	F16
10	F32

<Qd>   Is the 128-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field as <Qd>*2.

<Qm>   Is the 128-bit name of the SIMD&FP source register, encoded in the "M:Vm" field as <Qm>*2.

<Dd>   Is the 64-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field.

<Dm>   Is the 64-bit name of the SIMD&FP source register, encoded in the "M:Vm" field.

Operation

```
EncodingSpecificOperations(); CheckAdvSIMDEnabled();
bits(esize) result;
for r = 0 to regs-1
    for e = 0 to elements-1
        Elem[D[d+r],e,esize] = FPToFixed(Elem[D[m+r],e,esize], 0, unsigned,
                                        StandardFPSCRValue(), rounding);
```

(old)	htmldiff from-	(new)

VCVTP (Advanced SIMD)

Vector Convert floating-point to integer with Round towards +Infinity converts each element in a vector from floating-point to integer using the Round towards +Infinity rounding mode, and places the results in a second vector.

The operand vector elements are floating-point numbers.

The result vector elements are integers, 32-bit and the same size as the operand vector elements. integers. Signed and unsigned integers are distinct.

Depending on settings in the *CPACR*, *NSACR*, *HCPTR*, and *FPEXC* registers, and the Security state and PE mode in which the instruction is executed, an attempt to execute the instruction might be UNDEFINED, or trapped to Hyp mode. For more information see *Enabling Advanced SIMD and floating-point support*.

It has encodings from the following instruction sets: A32 (A1) and T32 (T1) .

A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	D	1	1	size		1	1	Vd				0	0	1	0	op	Q	M	0	Vm			

RM

64-bit SIMD vector (Q == 0)

VCVTP{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTP{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

T1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	D	1	1	size		1	1	Vd				0	0	1	0	op	Q	M	0	Vm		

RM

64-bit SIMD vector (Q == 0)

VCVTP{<q>}.<dt>.<dt2> <Dd>, <Dm>

128-bit SIMD vector (Q == 1)

VCVTP{<q>}.<dt>.<dt2> <Qd>, <Qm>

```
if InITBlock() then UNPREDICTABLE;
if Q == '1' && (Vd<0> == '1' || Vm<0> == '1') then UNDEFINED;
if (size == '01' && !HaveFP16Ext()) || size IN {'00', '11'} then UNDEFINED;
rounding = FPDecodeRM(RM);  unsigned = (op == '1');
case size of
    when '01' esize = 16; elements = 4;
    when '10' esize = 32; elements = 2;
d = UInt(D:Vd);  m = UInt(M:Vm);  regs = if Q == '0' then 1 else 2;
```

CONSTRAINED UNPREDICTABLE behavior

If `InITBlock()`, then one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction executes as if it passes the Condition code check.
- The instruction executes as NOP. This means it behaves as if it fails the Condition code check.

Assembler Symbols

<q>   See *Standard assembler syntax fields*.

<dt>   Is the data type for the elements of the destination, encoded in "op:size":

op	size	<dt>
0	01 S32	S16
0 1	10 U32	S32
1	01	U16
1	10	U32

<dt2>   Is the data type for the elements of the source vector, encoded in "size":

size	<dt2>
01	F16
10	F32

<Qd>   Is the 128-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field as <Qd>*2.

<Qm>   Is the 128-bit name of the SIMD&FP source register, encoded in the "M:Vm" field as <Qm>*2.

<Dd>   Is the 64-bit name of the SIMD&FP destination register, encoded in the "D:Vd" field.

<Dm>   Is the 64-bit name of the SIMD&FP source register, encoded in the "M:Vm" field.

Operation

```
EncodingSpecificOperations(); CheckAdvSIMDEnabled();
bits(esize) result;
for r = 0 to regs-1
    for e = 0 to elements-1
        Elem[D[d+r],e,esize] = FPToFixed(Elem[D[m+r],e,esize], 0, unsigned,
                                         StandardFPSCRValue(), rounding);
```

(old)   htmldiff from-   (new)

Top-level encodings for A32

31 30 29 28	27 26 25	24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
cond	op0		op1	

Decode fields

cond	op0	op1	Instruction details
!= 1111	00x		Data-processing and miscellaneous instructions
!= 1111	010		Load/Store Word, Unsigned Byte (immediate, literal)
!= 1111	011	0	Load/Store Word, Unsigned Byte (register)
!= 1111	011	1	Media instructions
	10x		Branch, branch with link, and block data transfer
	11x		System register access, Advanced SIMD, floating-point, and Supervisor call
1111	0xx		Unconditional instructions

Data-processing and miscellaneous instructions

These instructions are under the top-level.

31 30 29 28	27 26	25	24 23 22 21 20	19 18 17 16 15 14 13 12 11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	00	op0	op1		op2	op3	op4	

Decode fields

op0	op1	op2	op3	op4	Instruction details
0		1	!= 00	1	Extra load/store
0	0xxxx	1	00	1	Multiply and Accumulate
0	1xxxx	1	00	1	Synchronization primitives and Load-Acquire/Store-Release
0	10xx0	0			Miscellaneous
0	10xx0	1		0	Halfword Multiply and Accumulate
0	!= 10xx0			0	Data-processing register (immediate shift)
0	!= 10xx0	0		1	Data-processing register (register shift)
1					Data-processing immediate

Extra load/store

These instructions are under Data-processing and miscellaneous instructions.

31 30 29 28	27 26 25	24 23	22	21 20 19 18 17 16 15 14 13 12 11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	000		op0		1	!= 00	1	

Decode fields

op0	Instruction details
0	Load/Store Dual, Half, Signed Byte (register)
1	Load/Store Dual, Half, Signed Byte (immediate, literal)

Load/Store Dual, Half, Signed Byte (register)

These instructions are under Extra load/store.

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6 5	4	3 2 1 0
!= 1111	0	0	0	P	U	0	W	o1	Rn	Rt	(0)	(0)	(0)	(0)	1	!= 00	1	Rm
cond																op2		

The following constraints also apply to this encoding: cond != 1111 && op2 != 00 && cond != 1111 && op2 != 00

Decode fields				**Instruction Details**
P	**W**	**o1**	**op2**	
0	0	0	01	STRH (register) — post-indexed
0	0	0	10	LDRD (register) — post-indexed
0	0	0	11	STRD (register) — post-indexed
0	0	1	01	LDRH (register) — post-indexed
0	0	1	10	LDRSB (register) — post-indexed
0	0	1	11	LDRSH (register) — post-indexed
0	1	0	01	STRHT
0	1	0	10	UNALLOCATED
0	1	0	11	UNALLOCATED
0	1	1	01	LDRHT
0	1	1	10	LDRSBT
0	1	1	11	LDRSHT
1		0	01	STRH (register) — pre-indexed
1		0	10	LDRD (register) — pre-indexed
1		0	11	STRD (register) — pre-indexed
1		1	01	LDRH (register) — pre-indexed
1		1	10	LDRSB (register) — pre-indexed
1		1	11	LDRSH (register) — pre-indexed

Load/Store Dual, Half, Signed Byte (immediate, literal)

These instructions are under [Extra load/store](#).

31 30 29 28	27 26 25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	0 0 0	P	U	1	W	o1	Rn	Rt	imm4H	1	!= 00	1	imm4L
cond											op2		

The following constraints also apply to this encoding: cond != 1111 && op2 != 00 && cond != 1111 && op2 != 00

Decode fields				**Instruction Details**
P:W	**o1**	**Rn**	**op2**	
	0	1111	10	LDRD (literal)
!= 01	1	1111	01	LDRH (literal)
!= 01	1	1111	10	LDRSB (literal)
!= 01	1	1111	11	LDRSH (literal)
00	0	!= 1111	10	LDRD (immediate) — post-indexed
00	0		01	STRH (immediate) — post-indexed
00	0		11	STRD (immediate) — post-indexed
00	1	!= 1111	01	LDRH (immediate) — post-indexed
00	1	!= 1111	10	LDRSB (immediate) — post-indexed
00	1	!= 1111	11	LDRSH (immediate) — post-indexed
01	0	!= 1111	10	UNALLOCATED
01	0		01	STRHT
01	0		11	UNALLOCATED
01	1		01	LDRHT
01	1		10	LDRSBT
01	1		11	LDRSHT
10	0	!= 1111	10	LDRD (immediate) — offset

P:W	o1	Rn	op2	Instruction Details
10	0		01	STRH (immediate) — offset
10	0		11	STRD (immediate) — offset
10	1	!= 1111	01	LDRH (immediate) — offset
10	1	!= 1111	10	LDRSB (immediate) — offset
10	1	!= 1111	11	LDRSH (immediate) — offset
11	0	!= 1111	10	LDRD (immediate) — pre-indexed
11	0		01	STRH (immediate) — pre-indexed
11	0		11	STRD (immediate) — pre-indexed
11	1	!= 1111	01	LDRH (immediate) — pre-indexed
11	1	!= 1111	10	LDRSB (immediate) — pre-indexed
11	1	!= 1111	11	LDRSH (immediate) — pre-indexed

The "Decode fields" header spans o1, Rn, op2 columns.

Multiply and Accumulate

These instructions are under [Data-processing and miscellaneous instructions](#).

31 30 29 28	27 26 25 24	23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
!= 1111	0 0 0 0	opc	S	RdHi	RdLo	Rm	1 0 0 1	Rn
cond								

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

opc	S	Instruction Details
000		MUL, MULS
001		MLA, MLAS
010	0	UMAAL
010	1	UNALLOCATED
011	0	MLS
011	1	UNALLOCATED
100		UMULL, UMULLS
101		UMLAL, UMLALS
110		SMULL, SMULLS
111		SMLAL, SMLALS

The "Decode fields" header spans opc and S columns.

Synchronization primitives and Load-Acquire/Store-Release

These instructions are under [Data-processing and miscellaneous instructions](#).

31 30 29 28	27 26 25 24	23	22 21 20 19 18 17 16 15 14 13 12	11 10	9 8 7	6 5 4	3 2 1 0
!= 1111	0001	op0		11		1001	

op0	Instruction details
0	UNALLOCATED
1	[Load/Store Exclusive and Load-Acquire/Store-Release](#)

The "Decode fields" header spans the op0 column.

Load/Store Exclusive and Load-Acquire/Store-Release

These instructions are under [Synchronization primitives and Load-Acquire/Store-Release](#).

31 30 29 28	27 26 25 24 23	22 21	20	19 18 17 16	15 14 13 12	11	10	9	8	7 6 5 4	3 2 1 0
!= 1111	0 0 0 1 1	size	L	Rn	xRd	(1)	(1)	ex	ord	1 0 0 1	xRt
cond											

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields				Instruction Details
size	L	ex	ord	
00	0	0	0	STL
00	0	0	1	UNALLOCATED
00	0	1	0	STLEX
00	0	1	1	STREX
00	1	0	0	LDA
00	1	0	1	UNALLOCATED
00	1	1	0	LDAEX
00	1	1	1	LDREX
01	0	0		UNALLOCATED
01	0	1	0	STLEXD
01	0	1	1	STREXD
01	1	0		UNALLOCATED
01	1	1	0	LDAEXD
01	1	1	1	LDREXD
10	0	0	0	STLB
10	0	0	1	UNALLOCATED
10	0	1	0	STLEXB
10	0	1	1	STREXB
10	1	0	0	LDAB
10	1	0	1	UNALLOCATED
10	1	1	0	LDAEXB
10	1	1	1	LDREXB
11	0	0	0	STLH
11	0	0	1	UNALLOCATED
11	0	1	0	STLEXH
11	0	1	1	STREXH
11	1	0	0	LDAH
11	1	0	1	UNALLOCATED
11	1	1	0	LDAEXH
11	1	1	1	LDREXH

Miscellaneous

These instructions are under [Data-processing and miscellaneous instructions](#).

31 30 29 28	27 26 25 24	23	22 21	20	19 18 17 16 15 14 13 12 11 10 9 8	7	6 5 4	3 2 1 0
!= 1111	00010		op0	0		0	op1	

Decode fields		Instruction details
op0	op1	
00	001	UNALLOCATED
00	010	UNALLOCATED
00	011	UNALLOCATED
00	110	UNALLOCATED

01	001	BX
01	010	BXJ
01	011	BLX (register)
01	110	UNALLOCATED
10	001	UNALLOCATED
10	010	UNALLOCATED
10	011	UNALLOCATED
10	110	UNALLOCATED
11	001	CLZ
11	010	UNALLOCATED
11	011	UNALLOCATED
11	110	ERET
	111	Exception Generation
	000	Move special register (register)
	100	Cyclic Redundancy Check
	101	Integer Saturating Arithmetic

Exception Generation

These instructions are under Miscellaneous.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16 15 14 13 12 11 10 9 8	7	6	5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	0	imm12	0	1	1	1	imm4
cond													

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	HLT
01	BKPT
10	HVC
11	SMC

Move special register (register)

These instructions are under Miscellaneous.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	0	mask	Rd	(0)	(0)	B	m	0	0	0	0	Rn
cond																		

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	B	Instruction Details
x0	0	MRS
x0	1	MRS (Banked register)
x1	0	MSR (register)
x1	1	MSR (Banked register)

Cyclic Redundancy Check

These instructions are under Miscellaneous.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
!= 1111	0	0	0	1	0	sz	0	Rn	Rd	(0)	(0)	C	(0)	0	1	0	0	Rm
cond																		

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields		Instruction Details
sz	C	
00	0	CRC32 — CRC32B
00	1	CRC32C — CRC32CB
01	0	CRC32 — CRC32H
01	1	CRC32C — CRC32CH
10	0	CRC32 — CRC32W
10	1	CRC32C — CRC32CW
11		CONSTRAINED UNPREDICTABLE

The behavior of the CONSTRAINED UNPREDICTABLE encodings in this table is described in *CONSTRAINED UNPREDICTABLE behavior for A32 and T32 instruction encodings*

Integer Saturating Arithmetic

These instructions are under Miscellaneous.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	0	Rn	Rd	(0)	(0)	(0)	(0)	0	1	0	1	Rm
cond																		

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields	Instruction Details
opc	
00	QADD
01	QSUB
10	QDADD
11	QDSUB

Halfword Multiply and Accumulate

These instructions are under Data-processing and miscellaneous instructions.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	0	Rd	Ra	Rm	1	M	N	0	Rn
cond															

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
opc	M	N	
00			SMLABB, SMLABT, SMLATB, SMLATT
01	0	0	SMLAWB, SMLAWT — SMLAWB
01	0	1	SMULWB, SMULWT — SMULWB

Decode fields			Instruction Details
opc	M	N	
01	1	0	SMLAWB, SMLAWT — SMLAWT
01	1	1	SMULWB, SMULWT — SMULWT
10			SMLALBB, SMLALBT, SMLALTB, SMLALTT
11			SMULBB, SMULBT, SMULTB, SMULTT

Data-processing register (immediate shift)

These instructions are under Data-processing and miscellaneous instructions.

31 30 29 28	27 26 25	24 23	22 21	20	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
!= 1111	000	op0		op1		0	

The following constraints also apply to this encoding: op0:op1 != 100

Decode fields		Instruction details
op0	op1	
0x		Integer Data Processing (three register, immediate shift)
10	1	Integer Test and Compare (two register, immediate shift)
11		Logical Arithmetic (three register, immediate shift)

Integer Data Processing (three register, immediate shift)

These instructions are under Data-processing register (immediate shift).

31 30 29 28	27	26	25	24	23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8 7	6 5	4	3 2 1 0
!= 1111	0	0	0	0	opc	S	Rn	Rd	imm5	stype	0	Rm
cond												

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
opc	S	Rn	
000			AND, ANDS (register)
001			EOR, EORS (register)
010	0	!= 1101	SUB, SUBS (register) — SUB
010	0	1101	SUB, SUBS (SP minus register) — SUB
010	1	!= 1101	SUB, SUBS (register) — SUBS
010	1	1101	SUB, SUBS (SP minus register) — SUBS
011			RSB, RSBS (register)
100	0	!= 1101	ADD, ADDS (register) — ADD
100	0	1101	ADD, ADDS (SP plus register) — ADD
100	1	!= 1101	ADD, ADDS (register) — ADDS
100	1	1101	ADD, ADDS (SP plus register) — ADDS
101			ADC, ADCS (register)
110			SBC, SBCS (register)
111			RSC, RSCS (register)

Integer Test and Compare (two register, immediate shift)

These instructions are under Data-processing register (immediate shift).

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15	14	13	12	11 10 9 8 7	6 5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	1	Rn	(0)	(0)	(0)	(0)	imm5	stype	0	Rm
cond																

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	TST (register)
01	TEQ (register)
10	CMP (register)
11	CMN (register)

Logical Arithmetic (three register, immediate shift)

These instructions are under [Data-processing register (immediate shift)](#).

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11 10 9 8 7	6 5	4	3 2 1 0
!= 1111	0	0	0	1	1	opc	S	Rn	Rd	imm5	stype	0	Rm
cond													

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	ORR, ORRS (register)
01	MOV, MOVS (register)
10	BIC, BICS (register)
11	MVN, MVNS (register)

Data-processing register (register shift)

These instructions are under [Data-processing and miscellaneous instructions](#).

31 30 29 28	27 26 25	24	23 22 21	20	19 18 17 16 15 14 13 12 11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	000	op0		op1		0		1	

The following constraints also apply to this encoding: op0:op1 != 100

Decode fields op0	op1	Instruction details
0x		[Integer Data Processing (three register, register shift)](#)
10	1	[Integer Test and Compare (two register, register shift)](#)
11		[Logical Arithmetic (three register, register shift)](#)

Integer Data Processing (three register, register shift)

These instructions are under [Data-processing register (register shift)](#).

31 30 29 28	27	26	25	24	23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	0	0	0	0	opc	S	Rn	Rd	Rs	0	stype	1	Rm
cond													

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
000	AND, ANDS (register-shifted register)
001	EOR, EORS (register-shifted register)
010	SUB, SUBS (register-shifted register)
011	RSB, RSBS (register-shifted register)
100	ADD, ADDS (register-shifted register)
101	ADC, ADCS (register-shifted register)
110	SBC, SBCS (register-shifted register)
111	RSC, RSCS (register-shifted register)

Integer Test and Compare (two register, register shift)

These instructions are under [Data-processing register (register shift)](#).

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15	14	13	12	11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	0	0	0	1	0	opc	1	Rn	(0)	(0)	(0)	(0)	Rs	0	stype	1	Rm
cond																	

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	TST (register-shifted register)
01	TEQ (register-shifted register)
10	CMP (register-shifted register)
11	CMN (register-shifted register)

Logical Arithmetic (three register, register shift)

These instructions are under [Data-processing register (register shift)](#).

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7	6 5	4	3 2 1 0
!= 1111	0	0	0	1	1	opc	S	Rn	Rd	Rs	0	stype	1	Rm
cond														

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	ORR, ORRS (register-shifted register)
01	MOV, MOVS (register-shifted register)
10	BIC, BICS (register-shifted register)
11	MVN, MVNS (register-shifted register)

Data-processing immediate

These instructions are under [Data-processing and miscellaneous instructions](#).

31 30 29 28	27 26 25	24 23 22 21	20	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	001	op0	op1	

Decode fields op0	op1	Instruction details

0x		Integer Data Processing (two register and immediate)
10	00	Move Halfword (immediate)
10	10	Move Special Register and Hints (immediate)
10	x1	Integer Test and Compare (one register and immediate)
11		Logical Arithmetic (two register and immediate)

Integer Data Processing (two register and immediate)

These instructions are under Data-processing immediate.

31 30 29 28	27 26 25 24	23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0 0 1 0	opc	S	Rn	Rd	imm12
cond						

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
opc	S	Rn	
000			AND, ANDS (immediate)
001			EOR, EORS (immediate)
010	0	!= 11x1	SUB, SUBS (immediate) — SUB
010	0	1101	SUB, SUBS (SP minus immediate) — SUB
010	0	1111	ADR — A2
010	1	!= 1101	SUB, SUBS (immediate) — SUBS
010	1	1101	SUB, SUBS (SP minus immediate) — SUBS
011			RSB, RSBS (immediate)
100	0	!= 11x1	ADD, ADDS (immediate) — ADD
100	0	1101	ADD, ADDS (SP plus immediate) — ADD
100	0	1111	ADR — A1
100	1	!= 1101	ADD, ADDS (immediate) — ADDS
100	1	1101	ADD, ADDS (SP plus immediate) — ADDS
101			ADC, ADCS (immediate)
110			SBC, SBCS (immediate)
111			RSC, RSCS (immediate)

Move Halfword (immediate)

These instructions are under Data-processing immediate.

31 30 29 28	27 26 25 24	23	22	21 20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0 0 1 1	0	H	0 0	imm4	Rd	imm12
cond							

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields	Instruction Details
H	
0	MOV, MOVS (immediate)
1	MOVT

Move Special Register and Hints (immediate)

These instructions are under Data-processing immediate.

Top-level encodings for A32

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15	14	13	12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0	0	1	1	0	R	1	0	imm4	(1)	(1)	(1)	(1)	imm12
cond														

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields		Instruction Details	Architecture Version
R:imm4	imm12		
!= 00000		MSR (immediate)	-
00000	xxxx00000000	NOP	-
00000	xxxx00000001	YIELD	-
00000	xxxx00000010	WFE	-
00000	xxxx00000011	WFI	-
00000	xxxx00000100	SEV	-
00000	xxxx00000101	SEVL	-
00000	xxxx0000011x	Reserved hint, behaves as NOP	-
00000	xxxx00001xxx	Reserved hint, behaves as NOP	-
00000	xxxx00010000	ESB	Armv8.2
00000	xxxx00010001	Reserved hint, behaves as NOP	-
00000	xxxx00010010	TSB CSYNC	Armv8.4
00000	xxxx00010011	Reserved hint, behaves as NOP	-
00000	xxxx00010100	CSDB	-
00000	xxxx00010101	Reserved hint, behaves as NOP	-
00000	xxxx00011xxx	Reserved hint, behaves as NOP	-
00000	xxxx0001111x	Reserved hint, behaves as NOP	-
00000	xxxx001xxxxx	Reserved hint, behaves as NOP	-
00000	xxxx01xxxxxx	Reserved hint, behaves as NOP	-
00000	xxxx10xxxxxx	Reserved hint, behaves as NOP	-
00000	xxxx110xxxxx	Reserved hint, behaves as NOP	-
00000	xxxx1110xxxx	Reserved hint, behaves as NOP	-
00000	xxxx1111xxxx	DBG	-

Integer Test and Compare (one register and immediate)

These instructions are under Data-processing immediate.

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15	14	13	12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0	0	1	1	0	opc	1	Rn	(0)	(0)	(0)	(0)	imm12
cond													

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields	Instruction Details
opc	
00	TST (immediate)
01	TEQ (immediate)
10	CMP (immediate)
11	CMN (immediate)

Logical Arithmetic (two register and immediate)

These instructions are under Data-processing immediate.

31 30 29 28	27 26 25 24 23	22 21	20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0 0 1 1 1	opc	S	Rn	Rd	imm12
cond						

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc	Instruction Details
00	ORR, ORRS (immediate)
01	MOV, MOVS (immediate)
10	BIC, BICS (immediate)
11	MVN, MVNS (immediate)

Load/Store Word, Unsigned Byte (immediate, literal)

31 30 29 28	27 26 25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	0 1 0	P	U	o2	W	o1	Rn	Rt	imm12
cond									

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

P:W	o2	o1	Rn	Instruction Details
!= 01	0	1	1111	LDR (literal)
!= 01	1	1	1111	LDRB (literal)
00	0	0		STR (immediate) — post-indexed
00	0	1	!= 1111	LDR (immediate) — post-indexed
00	1	0		STRB (immediate) — post-indexed
00	1	1	!= 1111	LDRB (immediate) — post-indexed
01	0	0		STRT
01	0	1		LDRT
01	1	0		STRBT
01	1	1		LDRBT
10	0	0		STR (immediate) — offset
10	0	1	!= 1111	LDR (immediate) — offset
10	1	0		STRB (immediate) — offset
10	1	1	!= 1111	LDRB (immediate) — offset
11	0	0		STR (immediate) — pre-indexed
11	0	1	!= 1111	LDR (immediate) — pre-indexed
11	1	0		STRB (immediate) — pre-indexed
11	1	1	!= 1111	LDRB (immediate) — pre-indexed

Load/Store Word, Unsigned Byte (register)

31 30 29 28	27 26 25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8 7	6 5	4	3 2 1 0
!= 1111	0 1 1	P	U	o2	W	o1	Rn	Rt	imm5	stype	0	Rm
cond												

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields				Instruction Details
P	o2	W	o1	
0	0	0	0	STR (register) — post-indexed

P	o2	W	o1	Instruction Details
	Decode fields			
0	0	0	1	LDR (register) — post-indexed
0	0	1	0	STRT
0	0	1	1	LDRT
0	1	0	0	STRB (register) — post-indexed
0	1	0	1	LDRB (register) — post-indexed
0	1	1	0	STRBT
0	1	1	1	LDRBT
1	0		0	STR (register) — pre-indexed
1	0		1	LDR (register) — pre-indexed
1	1		0	STRB (register) — pre-indexed
1	1		1	LDRB (register) — pre-indexed

Media instructions

These instructions are under the top-level.

31 30 29 28	27 26 25	24 23 22 21 20	19 18 17 16 15 14 13 12 11 10 9 8	7 6 5	4	3 2 1 0
!= 1111	011	op0		op1	1	

op0	op1	Instruction details
	Decode fields	
00xxx		Parallel Arithmetic
01000	101	SEL
01000	001	UNALLOCATED
01000	xx0	PKHBT, PKHTB
01001	x01	UNALLOCATED
01001	xx0	UNALLOCATED
0110x	x01	UNALLOCATED
0110x	xx0	UNALLOCATED
01x10	001	Saturate 16-bit
01x10	101	UNALLOCATED
01x11	x01	Reverse Bit/Byte
01x1x	xx0	Saturate 32-bit
01xxx	111	UNALLOCATED
01xxx	011	Extend and Add
10xxx		Signed multiply, Divide
11000	000	Unsigned Sum of Absolute Differences
11000	100	UNALLOCATED
11001	x00	UNALLOCATED
1101x	x00	UNALLOCATED
110xx	111	UNALLOCATED
1110x	111	UNALLOCATED
1110x	x00	Bitfield Insert
11110	111	UNALLOCATED
11111	111	Permanently UNDEFINED
1111x	x00	UNALLOCATED
11x0x	x10	UNALLOCATED
11x1x	x10	Bitfield Extract

`11xxx`	`011`	UNALLOCATED
`11xxx`	`x01`	UNALLOCATED

Parallel Arithmetic

These instructions are under [Media instructions](#).

31 30 29 28	27 26 25 24 23	22 21 20	19 18 17 16	15 14 13 12	11	10	9	8	7	6 5	4	3 2 1 0
!= 1111	0 1 1 0 0	op1	Rn	Rd	(1)	(1)	(1)	(1)	B	op2	1	Rm
cond												

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
op1	B	op2	
`000`			UNALLOCATED
`001`	`0`	`00`	SADD16
`001`	`0`	`01`	SASX
`001`	`0`	`10`	SSAX
`001`	`0`	`11`	SSUB16
`001`	`1`	`00`	SADD8
`001`	`1`	`01`	UNALLOCATED
`001`	`1`	`10`	UNALLOCATED
`001`	`1`	`11`	SSUB8
`010`	`0`	`00`	QADD16
`010`	`0`	`01`	QASX
`010`	`0`	`10`	QSAX
`010`	`0`	`11`	QSUB16
`010`	`1`	`00`	QADD8
`010`	`1`	`01`	UNALLOCATED
`010`	`1`	`10`	UNALLOCATED
`010`	`1`	`11`	QSUB8
`011`	`0`	`00`	SHADD16
`011`	`0`	`01`	SHASX
`011`	`0`	`10`	SHSAX
`011`	`0`	`11`	SHSUB16
`011`	`1`	`00`	SHADD8
`011`	`1`	`01`	UNALLOCATED
`011`	`1`	`10`	UNALLOCATED
`011`	`1`	`11`	SHSUB8
`100`			UNALLOCATED
`101`	`0`	`00`	UADD16
`101`	`0`	`01`	UASX
`101`	`0`	`10`	USAX
`101`	`0`	`11`	USUB16
`101`	`1`	`00`	UADD8
`101`	`1`	`01`	UNALLOCATED
`101`	`1`	`10`	UNALLOCATED
`101`	`1`	`11`	USUB8
`110`	`0`	`00`	UQADD16

Decode fields			Instruction Details
op1	B	op2	
110	0	01	UQASX
110	0	10	UQSAX
110	0	11	UQSUB16
110	1	00	UQADD8
110	1	01	UNALLOCATED
110	1	10	UNALLOCATED
110	1	11	UQSUB8
111	0	00	UHADD16
111	0	01	UHASX
111	0	10	UHSAX
111	0	11	UHSUB16
111	1	00	UHADD8
111	1	01	UNALLOCATED
111	1	10	UNALLOCATED
111	1	11	UHSUB8

Saturate 16-bit

These instructions are under Media instructions.

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
!= 1111	0	1	1	0	1	U	1	0	sat_imm	Rd	(1)	(1)	(1)	(1)	0	0	1	1	Rn
cond																			

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields	Instruction Details
U	
0	SSAT16
1	USAT16

Reverse Bit/Byte

These instructions are under Media instructions.

31 30 29 28	27	26	25	24	23	22	21	20	19	18	17	16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
!= 1111	0	1	1	0	1	o1	1	1	(1)	(1)	(1)	(1)	Rd	(1)	(1)	(1)	(1)	o2	0	1	1	Rm
cond																						

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields		Instruction Details
o1	o2	
0	0	REV
0	1	REV16
1	0	RBIT
1	1	REVSH

Saturate 32-bit

These instructions are under Media instructions.

31 30 29 28	27	26	25	24	23	22	21	20 19 18 17 16	15 14 13 12	11 10 9 8 7	6 5	4	3 2 1 0
!= 1111	0	1	1	0	1	U	1	sat_imm	Rd	imm5	sh	0 1	Rn
cond													

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields U	Instruction Details
0	SSAT
1	USAT

Extend and Add

These instructions are under Media instructions.

31 30 29 28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11 10	9	8	7 6 5 4	3 2 1 0
!= 1111	0	1	1	0	1	U	op	Rn	Rd	rotate	(0)	(0)	0 1 1 1	Rm
cond														

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
U	op	Rn	
0	00	!= 1111	SXTAB16
0	00	1111	SXTB16
0	10	!= 1111	SXTAB
0	10	1111	SXTB
0	11	!= 1111	SXTAH
0	11	1111	SXTH
1	00	!= 1111	UXTAB16
1	00	1111	UXTB16
1	10	!= 1111	UXTAB
1	10	1111	UXTB
1	11	!= 1111	UXTAH
1	11	1111	UXTH

Signed multiply, Divide

These instructions are under Media instructions.

31 30 29 28	27	26	25	24	23	22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5	4	3 2 1 0
!= 1111	0	1	1	1	0	op1	Rd	Ra	Rm	op2	1	Rn
cond												

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields			Instruction Details
op1	Ra	op2	
000	!= 1111	000	SMLAD, SMLADX — SMLAD
000	!= 1111	001	SMLAD, SMLADX — SMLADX
000	!= 1111	010	SMLSD, SMLSDX — SMLSD
000	!= 1111	011	SMLSD, SMLSDX — SMLSDX
000		1xx	UNALLOCATED
000	1111	000	SMUAD, SMUADX — SMUAD

op1	Decode fields Ra	op2	Instruction Details
000	1111	001	SMUAD, SMUADX — SMUADX
000	1111	010	SMUSD, SMUSDX — SMUSD
000	1111	011	SMUSD, SMUSDX — SMUSDX
001		000	SDIV
001		!= 000	UNALLOCATED
010			UNALLOCATED
011		000	UDIV
011		!= 000	UNALLOCATED
100		000	SMLALD, SMLALDX — SMLALD
100		001	SMLALD, SMLALDX — SMLALDX
100		010	SMLSLD, SMLSLDX — SMLSLD
100		011	SMLSLD, SMLSLDX — SMLSLDX
100		1xx	UNALLOCATED
101	!= 1111	000	SMMLA, SMMLAR — SMMLA
101	!= 1111	001	SMMLA, SMMLAR — SMMLAR
101		01x	UNALLOCATED
101		10x	UNALLOCATED
101		110	SMMLS, SMMLSR — SMMLS
101		111	SMMLS, SMMLSR — SMMLSR
101	1111	000	SMMUL, SMMULR — SMMUL
101	1111	001	SMMUL, SMMULR — SMMULR
11x			UNALLOCATED

Unsigned Sum of Absolute Differences

These instructions are under [Media instructions](#).

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
!= 1111	0	1	1	1	1	0	0	0	Rd	Ra	Rm	0	0	0	1	Rn
cond																

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields Ra	Instruction Details
!= 1111	USADA8
1111	USAD8

Bitfield Insert

These instructions are under [Media instructions](#).

31 30 29 28	27	26	25	24	23	22	21	20 19 18 17 16	15 14 13 12	11 10 9 8 7	6	5	4	3 2 1 0
!= 1111	0	1	1	1	1	1	0	msb	Rd	lsb	0	0	1	Rn
cond														

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields Rn	Instruction Details
!= 1111	BFI

Decode fields Rn	Instruction Details
1111	BFC

Permanently UNDEFINED

These instructions are under Media instructions.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16 15 14 13 12 11 10 9 8	7	6	5	4	3 2 1 0
!= 1111				0	1	1	1	1	1	1	1	imm12	1	1	1	1	imm4
cond																	

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields cond	Instruction Details
0xxx	UNALLOCATED
10xx	UNALLOCATED
110x	UNALLOCATED
1110	UDF

Bitfield Extract

These instructions are under Media instructions.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8 7	6	5	4	3 2 1 0
!= 1111				0	1	1	1	1	U	1	widthm1	Rd	lsb	1	0	1	Rn
cond																		

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields U	Instruction Details
0	SBFX
1	UBFX

Branch, branch with link, and block data transfer

These instructions are under the top-level.

31 30 29 28	27 26	25	24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
cond	10	op0	

Decode fields cond	op0	Instruction details
1111	0	Exception Save/Restore
!= 1111	0	Load/Store Multiple
	1	Branch (immediate)

Exception Save/Restore

These instructions are under Branch, branch with link, and block data transfer.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12 11 10 9 8 7 6 5	4 3 2 1 0
1	1	1	1	1	0	0	P	U	S	W	L	Rn	op	mode

P	U	S	L	Instruction Details
		0	0	UNALLOCATED
0	0	0	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — Decrement After
0	0	1	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — Decrement After
0	1	0	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — Increment After
0	1	1	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — Increment After
1	0	0	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — Decrement Before
1	0	1	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — Decrement Before
		1	1	UNALLOCATED
1	1	0	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — Increment Before
1	1	1	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — Increment Before

Decode fields shown in header: P, U, S, L

Load/Store Multiple

These instructions are under [Branch, branch with link, and block data transfer](#).

31 30 29 28	27	26	25	24	23	22 21	20	19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
!= 1111	1	0	0	P	U	op	W	L	Rn	register_list

cond

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

P	U	op	L	register_list	Instruction Details
0	0	0	0		STMDA, STMED
0	0	0	1		LDMDA, LDMFA
0	1	0	0		STM, STMIA, STMEA
0	1	0	1		LDM, LDMIA, LDMFD
		1	0		STM (User registers)
1	0	0	0		STMDB, STMFD
1	0	0	1		LDMDB, LDMEA
		1	1	0xxxxxxxxxxxxxxx	LDM (User registers)
1	1	0	0		STMIB, STMFA
1	1	0	1		LDMIB, LDMED
		1	1	1xxxxxxxxxxxxxxx	LDM (exception return)

Decode fields shown in header: P, U, op, L, register_list

Branch (immediate)

These instructions are under [Branch, branch with link, and block data transfer](#).

31 30 29 28	27	26	25	24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
cond	1	0	1	H	imm24

cond	H	Instruction Details
!= 1111	0	B
!= 1111	1	BL, BLX (immediate) — A1
1111		BL, BLX (immediate) — A2

Decode fields shown in header: cond, H

System register access, Advanced SIMD, floating-point, and Supervisor call

These instructions are under the [top-level](#).

31 30 29 28	27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12	11 10 9	8 7 6 5	4	3 2 1 0
cond	11	op0		op1		op2	

Decode fields				Instruction details
cond	op0	op1	op2	
	0x	111		System register load/store and 64-bit move
	10	10x	0	Floating-point data-processing
	10	111	1	System register 32-bit move
	11			Supervisor call
1111	0x	1x0		Advanced SIMD three registers of the same length extension
1111	10	1x0		Advanced SIMD two registers and a scalar extension
!= 1111	0x	10x		Advanced SIMD load/store and 64-bit move
!= 1111	10	10x	1	Advanced SIMD and floating-point 32-bit move

System register load/store and 64-bit move

These instructions are under System register access, Advanced SIMD, floating-point, and Supervisor call.

31 30 29 28	27 26 25	24 23 22 21	20 19 18 17 16 15 14 13 12	11 10 9	8 7 6 5 4 3 2 1 0
	110	op0		111	

Decode fields	Instruction details
op0	
00x0	System register 64-bit move
!= 00x0	System register load/store

System register 64-bit move

These instructions are under System register load/store and 64-bit move.

31 30 29 28	27 26 25 24 23	22	21	20	19 18 17 16	15 14 13 12	11 10 9	8	7 6 5 4	3 2 1 0
cond	1 1 0 0 0	D	0	L	Rt2	Rt	1 1 1	cp15	opc1	CRm

Decode fields			Instruction Details
cond	D	L	
!= 1111	1	0	MCRR
!= 1111	1	1	MRRC
	0		UNALLOCATED
1111	1		UNALLOCATED

System register load/store

These instructions are under System register load/store and 64-bit move.

31 30 29 28	27 26 25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9	8	7 6 5 4 3 2 1 0
cond	1 1 0	P	U	D	W	L	Rn	CRd	1 1 1	cp15	imm8

The following constraints also apply to this encoding: P:U:D:W != 00x0

		Decode fields						Instruction Details
cond	P:U:W	D	L	Rn	CRd	cp15		
!= 1111	!= 000	0			!= 0101	0		UNALLOCATED
!= 1111	!= 000	0	1	1111	0101	0		LDC (literal)
!= 1111	!= 000					1		UNALLOCATED
!= 1111	!= 000	1			0101	0		UNALLOCATED

cond	P:U:W	D	L	Rn	CRd	cp15	Instruction Details
!= 1111	0x1	0	0		0101	0	STC — post-indexed
!= 1111	0x1	0	1	!= 1111	0101	0	LDC (immediate) — post-indexed
!= 1111	010	0	0		0101	0	STC — unindexed
!= 1111	010	0	1	!= 1111	0101	0	LDC (immediate) — unindexed
!= 1111	1x0	0	0		0101	0	STC — offset
!= 1111	1x0	0	1	!= 1111	0101	0	LDC (immediate) — offset
!= 1111	1x1	0	0		0101	0	STC — pre-indexed
!= 1111	1x1	0	1	!= 1111	0101	0	LDC (immediate) — pre-indexed
1111	!= 000						UNALLOCATED

(The column group header above reads **Decode fields**.)

Floating-point data-processing

These instructions are under [System register access, Advanced SIMD, floating-point, and Supervisor call](#).

31 30 29 28	27 26 25 24	23 22 21	20 19 18 17	16 15 14 13 12	11 10	9 8	7 6	5	4	3 2 1 0
cond	1110	op0	op1		10	op2	op3		0	

cond	op0	op1	op2	op3	Instruction details
1111	0xxx		!= 00	0	[Floating-point conditional select](#)
1111	1x00		!= 00		[Floating-point minNum/maxNum](#)
1111	1x11	0000	!= 00	1	[Floating-point extraction and insertion](#)
1111	1x11	1xxx	!= 00	1	[Floating-point directed convert to integer](#)
!= 1111	1x11			1	[Floating-point data-processing (two registers)](#)
!= 1111	1x11			0	[Floating-point move immediate](#)
!= 1111	!= 1x11				[Floating-point data-processing (three registers)](#)

(The column group header above reads **Decode fields**.)

Floating-point conditional select

These instructions are under [Floating-point data-processing](#).

31	30	29	28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11 10	9 8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	0	0	D	cc	Vn	Vd	1 0	!= 00	N	0	M	0	Vm

size

The following constraints also apply to this encoding: size != 00 && size != 00

cc	size	Instruction Details
00		VSELEQ, VSELGE, VSELGT, VSELVS — VSELEQ
01		VSELEQ, VSELGE, VSELGT, VSELVS — VSELVS
	01	UNALLOCATED
10		VSELEQ, VSELGE, VSELGT, VSELVS — VSELGE
11		VSELEQ, VSELGE, VSELGT, VSELVS — VSELGT

(The column group header above reads **Decode fields**.)

Floating-point minNum/maxNum

These instructions are under [Floating-point data-processing](#).

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10	9 8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	0	1	D	0	0	Vn	Vd	1 0	!= 00	N	op	M	0	Vm

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields		Instruction Details
size	op	
	0	VMAXNM
01		UNALLOCATED
	1	VMINNM

Floating-point extraction and insertion

These instructions are under [Floating-point data-processing](#).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 14 13 12	11	10	9 8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	0	1	D	1	1	0	0	0	0	Vd	1	0	!= 00	op	1	M	0	Vm

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields		Instruction Details	Architecture Version
size	op		
01		UNALLOCATED	-
10	0	VMOVX	Armv8.2
10	1	VINS	Armv8.2
11		UNALLOCATED	-

Floating-point directed convert to integer

These instructions are under [Floating-point data-processing](#).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17 16	15 14 13 12	11	10	9 8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	0	1	D	1	1	1	o1	RM	Vd	1	0	!= 00	op	1	M	0	Vm

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields			Instruction Details
o1	RM	size	
0	00		VRINTA (floating-point)
0	01		VRINTN (floating-point)
		01	UNALLOCATED
0	10		VRINTP (floating-point)
0	11		VRINTM (floating-point)
1	00		VCVTA (floating-point)
1	01		VCVTN (floating-point)
1	10		VCVTP (floating-point)
1	11		VCVTM (floating-point)

Floating-point data-processing (two registers)

These instructions are under [Floating-point data-processing](#).

31 30 29 28	27	26	25	24	23	22	21	20	19	18 17 16	15 14 13 12	11	10	9 8	7	6	5	4	3 2 1 0
!= 1111	1	1	1	0	1	D	1	1	o1	opc2	Vd	1	0	size	o3	1	M	0	Vm

cond

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields				Instruction Details	Architecture Version
o1	opc2	size	o3		
		00		UNALLOCATED	-
0	000	01	0	UNALLOCATED	-
0	000		1	VABS	-
0	000	10	0	VMOV (register) — single-precision scalar	-
0	000	11	0	VMOV (register) — double-precision scalar	-
0	001		0	VNEG	-
0	001		1	VSQRT	-
0	010		0	VCVTB — half-precision to double-precision	-
0	010	01		UNALLOCATED	-
0	010		1	VCVTT — half-precision to double-precision	-
0	011	01	0	VCVTB (BFloat16)	Armv8.6
0	011	01	1	VCVTT (BFloat16)	Armv8.6
0	011	10	0	VCVTB — single-precision to half-precision	-
0	011	10	1	VCVTT — single-precision to half-precision	-
0	011	11	0	VCVTB — double-precision to half-precision	-
0	011	11	1	VCVTT — double-precision to half-precision	-
0	100		0	VCMP — A1	-
0	100		1	VCMPE — A1	-
0	101		0	VCMP — A2	-
0	101		1	VCMPE — A2	-
0	110		0	VRINTR	-
0	110		1	VRINTZ (floating-point)	-
0	111		0	VRINTX (floating-point)	-
0	111	01	1	UNALLOCATED	-
0	111	10	1	VCVT (between double-precision and single-precision) — single-precision to double-precision	-
0	111	11	1	VCVT (between double-precision and single-precision) — double-precision to single-precision	-
1	000			VCVT (integer to floating-point, floating-point)	-
1	001	01		UNALLOCATED	-
1	001	10		UNALLOCATED	-
1	001	11	0	UNALLOCATED	-
1	001	11	1	VJCVT	Armv8.3
1	01x			VCVT (between floating-point and fixed-point, floating-point)	-
1	100		0	VCVTR	-
1	100		1	VCVT (floating-point to integer, floating-point)	-
1	101		0	VCVTR	-
1	101		1	VCVT (floating-point to integer, floating-point)	-
1	11x			VCVT (between floating-point and fixed-point, floating-point)	-

Floating-point move immediate

These instructions are under Floating-point data-processing.

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9 8	7	6	5	4	3 2 1 0
!= 1111	1	1	1	0	1	D	1	1	imm4H	Vd	1	0	size	(0)	0	(0)	0	imm4L
cond																		

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields size	Instruction Details	Architecture Version
00	UNALLOCATED	-
01	VMOV (immediate) — half-precision scalar	Armv8.2
10	VMOV (immediate) — single-precision scalar	-
11	VMOV (immediate) — double-precision scalar	-

Floating-point data-processing (three registers)

These instructions are under Floating-point data-processing.

31 30 29 28	27 26 25 24	23	22	21 20	19 18 17 16	15 14 13 12	11 10	9 8	7	6	5	4	3 2 1 0
!= 1111	1 1 1 0	o0	D	o1	Vn	Vd	1 0	size	N	o2	M	0	Vm
cond													

The following constraints also apply to this encoding: cond != 1111 && o0:D:o1 != 1x11 && cond != 1111

Decode fields o0:o1	size	o2	Instruction Details
!= 111	00		UNALLOCATED
000		0	VMLA (floating-point)
000		1	VMLS (floating-point)
001		0	VNMLS
001		1	VNMLA
010		0	VMUL (floating-point)
010		1	VNMUL
011		0	VADD (floating-point)
011		1	VSUB (floating-point)
100		0	VDIV
101		0	VFNMS
101		1	VFNMA
110		0	VFMA
110		1	VFMS

System register 32-bit move

These instructions are under System register access, Advanced SIMD, floating-point, and Supervisor call.

31 30 29 28	27 26 25 24	23 22 21	20	19 18 17 16	15 14 13 12	11 10 9	8	7 6 5	4	3 2 1 0
cond	1 1 1 0	opc1	L	CRn	Rt	1 1 1	cp15	opc2	1	CRm

Decode fields cond	L	Instruction Details
!= 1111	0	MCR
!= 1111	1	MRC
1111		UNALLOCATED

Supervisor call

These instructions are under System register access, Advanced SIMD, floating-point, and Supervisor call.

31 30 29 28	27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
cond	1111	

Decode fields cond	Instruction details
1111	UNALLOCATED
!= 1111	SVC

Advanced SIMD three registers of the same length extension

These instructions are under [System register access, Advanced SIMD, floating-point, and Supervisor call](#).

31	30	29	28	27	26	25	24 23	22	21 20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	0	op1	D	op2	Vn	Vd	1	op3	0	op4	N	Q	M	U	Vm

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
x1	0x	0	0	0	0	VCADD — 64-bit SIMD vector	Armv8.3
x1	0x	0	0	0	1	UNALLOCATED	-
x1	0x	0	0	1	0	VCADD — 128-bit SIMD vector	Armv8.3
x1	0x	0	0	1	1	UNALLOCATED	-
00	0x	0	0			UNALLOCATED	-
00	0x	0	1			UNALLOCATED	-
00	00	1	0	0	0	UNALLOCATED	-
00	00	1	0	0	1	UNALLOCATED	-
00	00	1	0	1	0	VMMLA	Armv8.6
00	00	1	0	1	1	UNALLOCATED	-
00	00	1	1	0	0	VDOT (vector) — 64-bit SIMD vector	Armv8.6
00	00	1	1	0	1	UNALLOCATED	-
00	00	1	1	1	0	VDOT (vector) — 128-bit SIMD vector	Armv8.6
00	00	1	1	1	1	UNALLOCATED	-
00	01	1	0			UNALLOCATED	-
00	01	1	1			UNALLOCATED	-
00	10	0	0		1	VFMAL (vector)	Armv8.2
00	10	0	1			UNALLOCATED	-
00	10	1	0	0		UNALLOCATED	-
00	10	1	0	1	0	VSMMLA	Armv8.6
00	10	1	0	1	1	VUMMLA	Armv8.6
00	10	1	1	0	0	VSDOT (vector) — 64-bit SIMD vector	Armv8.2
00	10	1	1	0	1	VUDOT (vector) — 64-bit SIMD vector	Armv8.2
00	10	1	1	1	0	VSDOT (vector) — 128-bit SIMD vector	Armv8.2
00	10	1	1	1	1	VUDOT (vector) — 128-bit SIMD vector	Armv8.2
00	11	0	0		1	VFMAB, VFMAT (BFloat16, vector)	Armv8.6
00	11	0	1			UNALLOCATED	-
00	11	1	0			UNALLOCATED	-
00	11	1	1			UNALLOCATED	-
01	10	0	0		1	VFMSL (vector)	Armv8.2
01	10	0	1			UNALLOCATED	-
01	10	1	0	0		UNALLOCATED	-
01	10	1	0	1	0	VUSMMLA	Armv8.6
01	10	1	0	1	1	UNALLOCATED	-
01	10	1	1	0	0	VUSDOT (vector) — 64-bit SIMD vector	Armv8.6
01	10	1	1		1	UNALLOCATED	-
01	10	1	1	1	0	VUSDOT (vector) — 128-bit SIMD vector	Armv8.6

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
01	11					UNALLOCATED	-
	1x	0	0		0	VCMLA	Armv8.3
10	11					UNALLOCATED	-
11	11					UNALLOCATED	-

The column headers above span: **Decode fields** (op1, op2, op3, op4), Q, U.

Advanced SIMD two registers and a scalar extension

These instructions are under [System register access, Advanced SIMD, floating-point, and Supervisor call](#).

31	30	29	28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	0	op1	D	op2	Vn	Vd	1	op3	0	op4	N	Q	M	U	Vm

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
0		0	0		0	VCMLA (by element) — half-precision scalar	Armv8.3
0	00	0	0		1	VFMAL (by scalar)	Armv8.2
0	00	0	1			UNALLOCATED	-
0	00	1	0			UNALLOCATED	-
0	00	1	1	0	0	VDOT (by element) — 64-bit SIMD vector	Armv8.6
0	00	1	1		1	UNALLOCATED	-
0	00	1	1	1	0	VDOT (by element) — 128-bit SIMD vector	Armv8.6
0	01	0	0		0	UNALLOCATED	-
0	01	0	0	0	1	VFMSL (by scalar) — 64-bit SIMD vector	Armv8.2
0	01	0	0	1	1	VFMSL (by scalar) — 128-bit SIMD vector	Armv8.2
0	01	0	1			UNALLOCATED	-
0	01	1	0			UNALLOCATED	-
0	10	0				UNALLOCATED	-
0	10	1	0			UNALLOCATED	-
0	10	1	1	0	0	VSDOT (by element) — 64-bit SIMD vector	Armv8.2
0	10	1	1	0	1	VUDOT (by element) — 64-bit SIMD vector	Armv8.2
0	10	1	1	1	0	VSDOT (by element) — 128-bit SIMD vector	Armv8.2
0	10	1	1	1	1	VUDOT (by element) — 128-bit SIMD vector	Armv8.2
0	11	0	0		0	UNALLOCATED	-
0	11	0	0		1	VFMAB, VFMAT (BFloat16, by scalar)	Armv8.6
0	11	0	1			UNALLOCATED	-
0	11	1				UNALLOCATED	-
1		0	0		0	VCMLA (by element) — single-precision scalar	Armv8.3
1	00	1	1	0	0	VUSDOT (by element) — 64-bit SIMD vector	Armv8.6
1	00	1	1	0	1	VSUDOT (by element) — 64-bit SIMD vector	Armv8.6
1	00	1	1	1	0	VUSDOT (by element) — 128-bit SIMD vector	Armv8.6
1	00	1	1	1	1	VSUDOT (by element) — 128-bit SIMD vector	Armv8.6
1		0	1			UNALLOCATED	-
1	01	1	1			UNALLOCATED	-
1	1x	1	1			UNALLOCATED	-
1		1	0			UNALLOCATED	-

Advanced SIMD load/store and 64-bit move

These instructions are under [System register access, Advanced SIMD, floating-point, and Supervisor call](#).

31 30 29 28	27 26 25	24 23 22 21	20 19 18 17 16 15 14 13 12	11 10	9 8 7 6 5 4 3 2 1 0
!= 1111	110	op0		10	

Decode fields op0	Instruction details
00x0	[Advanced SIMD and floating-point 64-bit move](#)
!= 00x0	[Advanced SIMD and floating-point load/store](#)

Advanced SIMD and floating-point 64-bit move

These instructions are under [Advanced SIMD load/store and 64-bit move](#).

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9 8	7 6	5	4	3 2 1 0
!= 1111	1	1	0	0	0	D	0	op	Rt2	Rt	1	0	size	opc2	M	o3	Vm
cond																	

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

D	op	size	opc2	o3	Instruction Details
0					UNALLOCATED
1				0	UNALLOCATED
1		0x	00	1	UNALLOCATED
1			01		UNALLOCATED
1	0	10	00	1	VMOV (between two general-purpose registers and two single-precision registers) — from general-purpose registers
1	0	11	00	1	VMOV (between two general-purpose registers and a doubleword floating-point register) — from general-purpose registers
1			1x		UNALLOCATED
1	1	10	00	1	VMOV (between two general-purpose registers and two single-precision registers) — to general-purpose registers
1	1	11	00	1	VMOV (between two general-purpose registers and a doubleword floating-point register) — to general-purpose registers

Advanced SIMD and floating-point load/store

These instructions are under [Advanced SIMD load/store and 64-bit move](#).

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9 8	7 6 5 4 3 2 1 0
!= 1111	1	1	0	P	U	D	W	L	Rn	Vd	1	0	size	imm8
cond														

The following constraints also apply to this encoding: cond != 1111 && P:U:D:W != 00x0 && cond != 1111

P	U	W	L	Rn	size	imm8	Instruction Details
0	0	1					UNALLOCATED
0	1				0x		UNALLOCATED
0	1		0		10		VSTM, VSTMDB, VSTMIA
0	1		0		11	xxxxxxx0	VSTM, VSTMDB, VSTMIA
0	1		0		11	xxxxxxx1	FSTMDBX, FSTMIAX — Increment After
0	1		1		10		VLDM, VLDMDB, VLDMIA

P	U	W	L	Rn	size	imm8	Instruction Details
				Decode fields			
0	1		1		11	xxxxxxx0	VLDM, VLDMDB, VLDMIA
0	1		1		11	xxxxxxx1	FLDM*X (FLDMDBX, FLDMIAX) — Increment After
1		0	0				VSTR
1		0			00		UNALLOCATED
1		0	1	!= 1111			VLDR (immediate)
1	0	1			0x		UNALLOCATED
1	0	1	0		10		VSTM, VSTMDB, VSTMIA
1	0	1	0		11	xxxxxxx0	VSTM, VSTMDB, VSTMIA
1	0	1	0		11	xxxxxxx1	FSTMDBX, FSTMIAX — Decrement Before
1	0	1	1		10		VLDM, VLDMDB, VLDMIA
1	0	1	1		11	xxxxxxx0	VLDM, VLDMDB, VLDMIA
1	0	1	1		11	xxxxxxx1	FLDM*X (FLDMDBX, FLDMIAX) — Decrement Before
1		0	1	1111			VLDR (literal)
1	1	1					UNALLOCATED

Advanced SIMD and floating-point 32-bit move

These instructions are under System register access, Advanced SIMD, floating-point, and Supervisor call.

31 30 29 28	27 26 25 24	23 22 21	20 19 18 17 16 15 14 13 12	11 10	9 8	7 6 5	4 3 2 1 0
!= 1111	1110	op0		10	op1		11111

op0	op1	Instruction details	Architecture version
Decode fields			
000	01	VMOV (between general-purpose register and half-precision)	Armv8.2
000	10	VMOV (between general-purpose register and single-precision)	-
111	10	Floating-point move special register	-
	11	Advanced SIMD 8/16/32-bit element move/duplicate	-

Floating-point move special register

These instructions are under Advanced SIMD and floating-point 32-bit move.

31 30 29 28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0
!= 1111	1	1	1	0	1	1	1	L	reg	Rt	1	0	1	0	(0)	(0)	(0)	1	(0)	(0)	(0)	(0)
cond																						

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

L	Instruction Details
Decode fields	
0	VMSR
1	VMRS

Advanced SIMD 8/16/32-bit element move/duplicate

These instructions are under Advanced SIMD and floating-point 32-bit move.

31 30 29 28	27	26	25	24	23 22 21	20	19 18 17 16	15 14 13 12	11	10	9	8	7	6 5	4	3	2	1	0
!= 1111	1	1	1	0	opc1	L	Vn	Rt	1	0	1	1	N	opc2	1	(0)	(0)	(0)	(0)
cond																			

The following constraints also apply to this encoding: cond != 1111 && cond != 1111

Decode fields opc1	L	opc2	Instruction Details
0xx	0		VMOV (general-purpose register to scalar)
	1		VMOV (scalar to general-purpose register)
1xx	0	0x	VDUP (general-purpose register)
1xx	0	1x	UNALLOCATED

Unconditional instructions

These instructions are under the [top-level](#).

31 30 29 28 27	26 25	24 23 22 21	20	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
11110	op0		op1	

Decode fields op0	op1	Instruction details
00		[Miscellaneous](#)
01		[Advanced SIMD data-processing](#)
1x	1	[Memory hints and barriers](#)
10	0	[Advanced SIMD element or structure load/store](#)
11	0	UNALLOCATED

Miscellaneous

These instructions are under [Unconditional instructions](#).

31 30 29 28 27 26 25	24 23 22 21 20	19 18 17 16 15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
1111000	op0		op1	

Decode fields op0	op1	Instruction details	Architecture version
0xxxx		UNALLOCATED	-
10000	xx0x	[Change Process State](#)	-
10001	1000	UNALLOCATED	-
10001	x100	UNALLOCATED	-
10001	xx01	UNALLOCATED	-
10001	0000	SETPAN	Armv8.1
1000x	0111	UNALLOCATED	-
10010	0111	CONSTRAINED UNPREDICTABLE	-
10011	0111	UNALLOCATED	-
1001x	xx0x	UNALLOCATED	-
100xx	0011	UNALLOCATED	-
100xx	0x10	UNALLOCATED	-
100xx	1x1x	UNALLOCATED	-
101xx		UNALLOCATED	-
11xxx		UNALLOCATED	-

The behavior of the CONSTRAINED UNPREDICTABLE encodings in this table is described in *CONSTRAINED UNPREDICTABLE behavior for A32 and T32 instruction encodings*

Change Process State

These instructions are under [Miscellaneous](#).

31	30	29	28	27	26	25	24	23	22	21	20	19 18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	0	1	0	0	0	0	imod	M	op	(0)	(0)	(0)	(0)	(0)	(0)	E	A	I	F	0		mode

	Decode fields			Instruction Details
imod	M	op	mode	
		1	0xxxx	SETEND
		0		CPS, CPSID, CPSIE
		1	1xxxx	UNALLOCATED

Advanced SIMD data-processing

These instructions are under [Unconditional instructions](#).

31 30 29 28 27 26 25	24	23	22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
1111001		op0		op1	

Decode fields		Instruction details
op0	op1	
0		[Advanced SIMD three registers of the same length](#)
1	0	[Advanced SIMD two registers, or three registers of different lengths](#)
1	1	[Advanced SIMD shifts and immediate generation](#)

Advanced SIMD three registers of the same length

These instructions are under [Advanced SIMD data-processing](#).

31	30	29	28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	U	0	D	size	Vn	Vd	opc	N	Q	M	o1	Vm

	Decode fields				Instruction Details	Architecture Version
U	size	opc	Q	o1		
0	0x	1100		1	VFMA	-
0	0x	1101		0	VADD (floating-point)	-
0	0x	1101		1	VMLA (floating-point)	-
0	0x	1110		0	VCEQ (register) — A2	-
0	0x	1111		0	VMAX (floating-point)	-
0	0x	1111		1	VRECPS	-
		0000		0	VHADD	-
0	00	0001		1	VAND (register)	-
		0000		1	VQADD	-
		0001		0	VRHADD	-
0	00	1100		0	SHA1C	-
		0010		0	VHSUB	-
0	01	0001		1	VBIC (register)	-
		0010		1	VQSUB	-
		0011		0	VCGT (register) — A1	-
		0011		1	VCGE (register) — A1	-
0	01	1100		0	SHA1P	-
0	1x	1100		1	VFMS	-
0	1x	1101		0	VSUB (floating-point)	-

	Decode fields				Instruction Details	Architecture Version
U	size	opc	Q	o1		
0	1x	1101		1	VMLS (floating-point)	-
0	1x	1110		0	UNALLOCATED	-
0	1x	1111		0	VMIN (floating-point)	-
0	1x	1111		1	VRSQRTS	-
		0100		0	VSHL (register)	-
0		1000		0	VADD (integer)	-
0	10	0001		1	VORR (register)	-
0		1000		1	VTST	-
		0100		1	VQSHL (register)	-
0		1001		0	VMLA (integer)	-
		0101		0	VRSHL	-
		0101		1	VQRSHL	-
0		1011		0	VQDMULH	-
0	10	1100		0	SHA1M	-
0		1011		1	VPADD (integer)	-
		0110		0	VMAX (integer)	-
0	11	0001		1	VORN (register)	-
		0110		1	VMIN (integer)	-
		0111		0	VABD (integer)	-
		0111		1	VABA	-
0	11	1100		0	SHA1SU0	-
1	0x	1101		0	VPADD (floating-point)	-
1	0x	1101		1	VMUL (floating-point)	-
1	0x	1110		0	VCGE (register) — A2	-
1	0x	1110		1	VACGE	-
1	0x	1111	0	0	VPMAX (floating-point)	-
1	0x	1111		1	VMAXNM	-
1	00	0001		1	VEOR	-
		1001		1	VMUL (integer and polynomial)	-
1	00	1100		0	SHA256H	-
		1010	0	0	VPMAX (integer)	-
1	01	0001		1	VBSL	-
		1010	0	1	VPMIN (integer)	-
		1010	1		UNALLOCATED	-
1	01	1100		0	SHA256H2	-
1	1x	1101		0	VABD (floating-point)	-
1	1x	1110		0	VCGT (register) — A2	-
1	1x	1110		1	VACGT	-
1	1x	1111	0	0	VPMIN (floating-point)	-
1	1x	1111		1	VMINNM	-
1		1000		0	VSUB (integer)	-
1	10	0001		1	VBIT	-
1		1000		1	VCEQ (register) — A1	-
1		1001		0	VMLS (integer)	-
1		1011		0	VQRDMULH	-
1	10	1100		0	SHA256SU1	-
1		1011		1	VQRDMLAH	Armv8.1

	Decode fields				Instruction Details	Architecture Version
U	size	opc	Q	o1		
1	11	0001		1	VBIF	-
1		1100		1	VQRDMLSH	Armv8.1
1		1111	1	0	UNALLOCATED	-

Advanced SIMD two registers, or three registers of different lengths

These instructions are under Advanced SIMD data-processing.

31 30 29 28 27 26 25	24	23 22	21 20	19 18 17 16 15 14 13 12	11 10 9 8 7	6 5	4	3 2 1 0
1111001	op0	1	op1		op2	op3	0	

	Decode fields			Instruction details
op0	op1	op2	op3	
0	11			VEXT (byte elements)
1	11	0x		Advanced SIMD two registers misc
1	11	10		VTBL, VTBX
1	11	11		Advanced SIMD duplicate (scalar)
	!= 11		0	Advanced SIMD three registers of different lengths
	!= 11		1	Advanced SIMD two registers and a scalar

Advanced SIMD two registers misc

These instructions are under Advanced SIMD two registers, or three registers of different lengths.

31	30	29	28	27	26	25	24	23	22	21	20	19 18	17 16	15 14 13 12	11	10 9 8 7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	1	1	D	1	1	size	opc1	Vd	0	opc2	Q	M	0	Vm

	Decode fields			Instruction Details	Architecture Version
size	opc1	opc2	Q		
	00	0000		VREV64	-
	00	0001		VREV32	-
	00	0010		VREV16	-
	00	0011		UNALLOCATED	-
	00	010x		VPADDL	-
	00	0110	0	AESE	-
	00	0110	1	AESD	-
	00	0111	0	AESMC	-
	00	0111	1	AESIMC	-
	00	1000		VCLS	-
00	10	0000		VSWP	-
	00	1001		VCLZ	-
	00	1010		VCNT	-
	00	1011		VMVN (register)	-
00	10	1100	1	UNALLOCATED	-
	00	110x		VPADAL	-
	00	1110		VQABS	-
	00	1111		VQNEG	-
	01	x000		VCGT (immediate #0)	-
	01	x001		VCGE (immediate #0)	-
	01	x010		VCEQ (immediate #0)	-

size	Decode fields opc1	opc2	Q	Instruction Details	Architecture Version
	01	x011		VCLE (immediate #0)	-
	01	x100		VCLT (immediate #0)	-
	01	x110		VABS	-
	01	x111		VNEG	-
	01	0101	1	SHA1H	-
01	10	1100	1	VCVT (from single-precision to BFloat16, Advanced SIMD)	Armv8.6
	10	0001		VTRN	-
	10	0010		VUZP	-
	10	0011		VZIP	-
	10	0100	0	VMOVN	-
	10	0100	1	VQMOVN, VQMOVUN — VQMOVUN	-
	10	0101		VQMOVN, VQMOVUN — VQMOVN	-
	10	0110	0	VSHLL	-
	10	0111	0	SHA1SU1	-
	10	0111	1	SHA256SU0	-
	10	1000		VRINTN (Advanced SIMD)	-
	10	1001		VRINTX (Advanced SIMD)	-
	10	1010		VRINTA (Advanced SIMD)	-
	10	1011		VRINTZ (Advanced SIMD)	-
10	10	1100	1	UNALLOCATED	-
	10	1100	0	VCVT (between half-precision and single-precision, Advanced SIMD) — single-precision to half-precision	-
	10	1101		VRINTM (Advanced SIMD)	-
	10	1110	0	VCVT (between half-precision and single-precision, Advanced SIMD) — half-precision to single-precision	-
	10	1110	1	UNALLOCATED	-
	10	1111		VRINTP (Advanced SIMD)	-
	11	000x		VCVTA (Advanced SIMD)	-
	11	001x		VCVTN (Advanced SIMD)	-
	11	010x		VCVTP (Advanced SIMD)	-
	11	011x		VCVTM (Advanced SIMD)	-
	11	10x0		VRECPE	-
	11	10x1		VRSQRTE	-
11	10	1100	1	UNALLOCATED	-
	11	11xx		VCVT (between floating-point and integer, Advanced SIMD)	-

Advanced SIMD duplicate (scalar)

These instructions are under Advanced SIMD two registers, or three registers of different lengths.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	D	1	1	imm4					Vd				1	1	opc			Q	M	0	Vm			

Decode fields opc	Instruction Details
000	VDUP (scalar)
001	UNALLOCATED
01x	UNALLOCATED
1xx	UNALLOCATED

Advanced SIMD three registers of different lengths

These instructions are under [Advanced SIMD two registers, or three registers of different lengths](#).

31	30	29	28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	U	1	D	!= 11	Vn	Vd	opc	N	0	M	0	Vm

size

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields U	opc	Instruction Details
	0000	VADDL
	0001	VADDW
	0010	VSUBL
0	0100	VADDHN
	0011	VSUBW
0	0110	VSUBHN
0	1001	VQDMLAL
	0101	VABAL
0	1011	VQDMLSL
0	1101	VQDMULL
	0111	VABDL (integer)
	1000	VMLAL (integer)
	1010	VMLSL (integer)
1	0100	VRADDHN
1	0110	VRSUBHN
	11x0	VMULL (integer and polynomial)
1	1001	UNALLOCATED
1	1011	UNALLOCATED
1	1101	UNALLOCATED
	1111	UNALLOCATED

Advanced SIMD two registers and a scalar

These instructions are under [Advanced SIMD two registers, or three registers of different lengths](#).

31	30	29	28	27	26	25	24	23	22	21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	Q	1	D	!= 11	Vn	Vd	opc	N	1	M	0	Vm

size

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields Q	opc	Instruction Details	Architecture Version
	000x	VMLA (by scalar)	-
0	0011	VQDMLAL	-
	0010	VMLAL (by scalar)	-
0	0111	VQDMLSL	-
	010x	VMLS (by scalar)	-
0	1011	VQDMULL	-
	0110	VMLSL (by scalar)	-
	100x	VMUL (by scalar)	-

Decode fields		Instruction Details	Architecture Version
Q	opc		
1	0011	UNALLOCATED	-
	1010	VMULL (by scalar)	-
1	0111	UNALLOCATED	-
	1100	VQDMULH	-
	1101	VQRDMULH	-
1	1011	UNALLOCATED	-
	1110	VQRDMLAH	Armv8.1
	1111	VQRDMLSH	Armv8.1

Advanced SIMD shifts and immediate generation

These instructions are under [Advanced SIMD data-processing](#).

31 30 29 28 27 26 25	24	23	22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
1111001		1	op0	1	

Decode fields op0	Instruction details
000xxxxxxxxxxx0	[Advanced SIMD one register and modified immediate](#)
!= 000xxxxxxxxxxx0	[Advanced SIMD two registers and shift amount](#)

Advanced SIMD one register and modified immediate

These instructions are under [Advanced SIMD shifts and immediate generation](#).

31	30	29	28	27	26	25	24	23	22	21	20	19	18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	i	1	D	0	0	0	imm3	Vd	cmode	0	Q	op	1	imm4

Decode fields		Instruction Details
cmode	op	
0xx0	0	VMOV (immediate) — A1
0xx0	1	VMVN (immediate) — A1
0xx1	0	VORR (immediate) — A1
0xx1	1	VBIC (immediate) — A1
10x0	0	VMOV (immediate) — A3
10x0	1	VMVN (immediate) — A2
10x1	0	VORR (immediate) — A2
10x1	1	VBIC (immediate) — A2
11xx	0	VMOV (immediate) — A4
110x	1	VMVN (immediate) — A3
1110	1	VMOV (immediate) — A5
1111	1	UNALLOCATED

Advanced SIMD two registers and shift amount

These instructions are under [Advanced SIMD shifts and immediate generation](#).

31	30	29	28	27	26	25	24	23	22	21 20 19	18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	0	0	1	U	1	D	imm3H	imm3L	Vd	opc	L	Q	M	1	Vm

The following constraints also apply to this encoding: imm3H:imm3L:Vd:opc:L != 000xxxxxxxxxxx0

U	imm3H:L	imm3L	opc	Q	Instruction Details
	!= 0000		0000		VSHR
	!= 0000		0001		VSRA
	!= 0000	000	1010	0	VMOVL
	!= 0000		0010		VRSHR
	!= 0000		0011		VRSRA
	!= 0000		0111		VQSHL, VQSHLU (immediate) — VQSHL
	!= 0000		1001	0	VQSHRN, VQSHRUN — VQSHRN
	!= 0000		1001	1	VQRSHRN, VQRSHRUN — VQRSHRN
	!= 0000		1010	0	VSHLL
	!= 0000		11xx		VCVT (between floating-point and fixed-point, Advanced SIMD)
0	!= 0000		0101		VSHL (immediate)
0	!= 0000		1000	0	VSHRN
0	!= 0000		1000	1	VRSHRN
1	!= 0000		0100		VSRI
1	!= 0000		0101		VSLI
1	!= 0000		0110		VQSHL, VQSHLU (immediate) — VQSHLU
1	!= 0000		1000	0	VQSHRN, VQSHRUN — VQSHRUN
1	!= 0000		1000	1	VQRSHRN, VQRSHRUN — VQRSHRUN

The top of the table header reads: **Decode fields** (spanning imm3H:L, imm3L, opc, Q columns).

Memory hints and barriers

These instructions are under Unconditional instructions.

31 30 29 28 27 26	25 24 23 22 21	20	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
111101	op0	1		op1	

Decode fields op0	op1	Instruction details
00xx1		CONSTRAINED UNPREDICTABLE
01001		CONSTRAINED UNPREDICTABLE
01011		Barriers
011x1		CONSTRAINED UNPREDICTABLE
0xxx0		Preload (immediate)
1xxx0	0	Preload (register)
1xxx1	0	CONSTRAINED UNPREDICTABLE
1xxxx	1	UNALLOCATED

The behavior of the CONSTRAINED UNPREDICTABLE encodings in this table is described in *CONSTRAINED UNPREDICTABLE behavior for A32 and T32 instruction encodings*

Barriers

These instructions are under Memory hints and barriers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7 6 5 4	3 2 1 0
1	1	1	1	0	1	0	1	0	1	1	1	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	(0)	(0)	opcode	option

Decode fields opcode	option	Instruction Details
0000		CONSTRAINED UNPREDICTABLE

opcode	option	Instruction Details
0001		CLREX
001x		CONSTRAINED UNPREDICTABLE
0100	!= 0x00	DSB
0100	0000	SSBB
0100	0100	PSSBB
0101		DMB
0110		ISB
0111		SB
1xxx		CONSTRAINED UNPREDICTABLE

The table above has header: **Decode fields** — opcode, option — and **Instruction Details**.

The behavior of the CONSTRAINED UNPREDICTABLE encodings in this table is described in *CONSTRAINED UNPREDICTABLE behavior for A32 and T32 instruction encodings*

Preload (immediate)

These instructions are under Memory hints and barriers.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15	14	13	12	11 10 9 8 7 6 5 4 3 2 1 0
1	1	1	1	0	1	0	D	U	R	0	1	Rn	(1)	(1)	(1)	(1)	imm12

D	R	Rn	Instruction Details
0	0		Reserved hint, behaves as NOP
0	1		PLI (immediate, literal)
1		1111	PLD (literal)
1	0	!= 1111	PLD, PLDW (immediate) — preload write
1	1	!= 1111	PLD, PLDW (immediate) — preload read

(Header: **Decode fields** — D, R, Rn — and **Instruction Details**.)

Preload (register)

These instructions are under Memory hints and barriers.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15	14	13	12	11 10 9 8 7	6 5 4	3	2 1 0
1	1	1	1	0	1	1	D	U	o2	0	1	Rn	(1)	(1)	(1)	(1)	imm5	stype	0	Rm

D	o2	Instruction Details
0	0	Reserved hint, behaves as NOP
0	1	PLI (register)
1	0	PLD, PLDW (register) — preload write
1	1	PLD, PLDW (register) — preload read

(Header: **Decode fields** — D, o2 — and **Instruction Details**.)

Advanced SIMD element or structure load/store

These instructions are under Unconditional instructions.

31 30 29 28 27 26 25 24	23 22 21	20	19 18 17 16 15 14 13 12	11 10	9 8 7 6 5 4 3 2 1 0
11110100	op0	0		op1	

op0	op1	Instruction details
0		Advanced SIMD load/store multiple structures

(Header: **Decode fields** — op0, op1 — and **Instruction details**.)

1	11	Advanced SIMD load single structure to all lanes
1	!= 11	Advanced SIMD load/store single structure to one lane

Advanced SIMD load/store multiple structures

These instructions are under Advanced SIMD element or structure load/store.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6	5 4	3 2 1 0
1	1	1	1	0	1	0	0	0	D	L	0	Rn	Vd	itype	size	align	Rm

Decode fields L	itype	Instruction Details
0	000x	VST4 (multiple 4-element structures)
0	0010	VST1 (multiple single elements) — A4
0	0011	VST2 (multiple 2-element structures) — A2
0	010x	VST3 (multiple 3-element structures)
0	0110	VST1 (multiple single elements) — A3
0	0111	VST1 (multiple single elements) — A1
0	100x	VST2 (multiple 2-element structures) — A1
0	1010	VST1 (multiple single elements) — A2
1	000x	VLD4 (multiple 4-element structures)
1	0010	VLD1 (multiple single elements) — A4
1	0011	VLD2 (multiple 2-element structures) — A2
1	010x	VLD3 (multiple 3-element structures)
	1011	UNALLOCATED
1	0110	VLD1 (multiple single elements) — A3
1	0111	VLD1 (multiple single elements) — A1
	11xx	UNALLOCATED
1	100x	VLD2 (multiple 2-element structures) — A1
1	1010	VLD1 (multiple single elements) — A2

Advanced SIMD load single structure to all lanes

These instructions are under Advanced SIMD element or structure load/store.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10	9 8	7 6	5	4	3 2 1 0
1	1	1	1	0	1	0	0	1	D	L	0	Rn	Vd	1 1	N	size	T	a	Rm

Decode fields L	N	a	Instruction Details
0			UNALLOCATED
1	00		VLD1 (single element to all lanes)
1	01		VLD2 (single 2-element structure to all lanes)
1	10	0	VLD3 (single 3-element structure to all lanes)
1	10	1	UNALLOCATED
1	11		VLD4 (single 4-element structure to all lanes)

Advanced SIMD load/store single structure to one lane

These instructions are under Advanced SIMD element or structure load/store.

31	30	29	28	27	26	25	24	23	22	21	20	19 18 17 16	15 14 13 12	11 10	9 8	7 6 5 4	3 2 1 0
1	1	1	1	0	1	0	0	1	D	L	0	Rn	Vd	!= 11	N	index_align	Rm
														size			

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields / Instruction Details

L	size	N	Rm	Instruction Details
0	00	00		VST1 (single element from one lane) — A1
0	00	01		VST2 (single 2-element structure from one lane) — A1
0	00	10		VST3 (single 3-element structure from one lane) — A1
0	00	11		VST4 (single 4-element structure from one lane)
0	01	00		VST1 (single element from one lane) — A2
0	01	01		VST2 (single 2-element structure from one lane) — A2
0	01	10		VST3 (single 3-element structure from one lane) — A2
0	01	11		VST4 (single 4-element structure from one lane)
0	10	00		VST1 (single element from one lane) — A3
0	10	01		VST2 (single 2-element structure from one lane) — A3
0	10	10		VST3 (single 3-element structure from one lane) — A3
0	10	11	!= 11x1	VST4 (single 4-element structure from one lane)
0	10	11	1101	VST4 (single 4-element structure from one lane)
0	10	11	1111	VST4 (single 4-element structure from one lane)
1	00	00		VLD1 (single element to one lane) — A1
1	00	01		VLD2 (single 2-element structure to one lane) — A1
1	00	10		VLD3 (single 3-element structure to one lane) — A1
1	00	11		VLD4 (single 4-element structure to one lane) — A1
1	01	00		VLD1 (single element to one lane) — A2
1	01	01		VLD2 (single 2-element structure to one lane) — A2
1	01	10		VLD3 (single 3-element structure to one lane) — A2
1	01	11		VLD4 (single 4-element structure to one lane) — A2
1	10	00		VLD1 (single element to one lane) — A3
1	10	01		VLD2 (single 2-element structure to one lane) — A3
1	10	10		VLD3 (single 3-element structure to one lane) — A3
1	10	11		VLD4 (single 4-element structure to one lane) — A3

(old) **htmldiff from-** **(new)**

Top-level encodings for T32

15 14 13	12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
op0 op1	

Decode fields		Instruction details
op0	op1	
!= 111		16-bit
111	00	B — T2
111	!= 00	32-bit

16-bit

These instructions are under the top-level.

15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
op0	

The following constraints also apply to this encoding: op0<5:3> != 111

Decode fields	Instruction details
op0	
00xxxx	Shift (immediate), add, subtract, move, and compare
010000	Data-processing (two low registers)
010001	Special data instructions and branch and exchange
01001x	LDR (literal) — T1
0101xx	Load/store (register offset)
011xxx	Load/store word/byte (immediate offset)
1000xx	Load/store halfword (immediate offset)
1001xx	Load/store (SP-relative)
1010xx	Add PC/SP (immediate)
1011xx	Miscellaneous 16-bit instructions
1100xx	Load/store multiple
1101xx	Conditional branch, and Supervisor Call

Shift (immediate), add, subtract, move, and compare

These instructions are under 16-bit.

15 14	13 12 11	10	9 8 7 6 5 4 3 2 1 0
00	op0 op1	op2	

Decode fields			Instruction details
op0	op1	op2	
0	11	0	Add, subtract (three low registers)
0	11	1	Add, subtract (two low registers and immediate)
0	!= 11		MOV, MOVS (register) — T2
1			Add, subtract, compare, move (one low register and immediate)

Page 69

Add, subtract (three low registers)

These instructions are under [Shift (immediate), add, subtract, move, and compare](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	S		Rm			Rn			Rd	

Decode fields S	Instruction Details
0	ADD, ADDS (register)
1	SUB, SUBS (register)

Add, subtract (two low registers and immediate)

These instructions are under [Shift (immediate), add, subtract, move, and compare](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	S		imm3			Rn			Rd	

Decode fields S	Instruction Details
0	ADD, ADDS (immediate)
1	SUB, SUBS (immediate)

Add, subtract, compare, move (one low register and immediate)

These instructions are under [Shift (immediate), add, subtract, move, and compare](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	op		Rd			imm8							

Decode fields op	Instruction Details
00	MOV, MOVS (immediate)
01	CMP (immediate)
10	ADD, ADDS (immediate)
11	SUB, SUBS (immediate)

Data-processing (two low registers)

These instructions are under [16-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0		op				Rs			Rd	

Decode fields op	Instruction Details
0000	AND, ANDS (register)
0001	EOR, EORS (register)
0010	MOV, MOVS (register-shifted register) — logical shift left
0011	MOV, MOVS (register-shifted register) — logical shift right
0100	MOV, MOVS (register-shifted register) — arithmetic shift right
0101	ADC, ADCS (register)
0110	SBC, SBCS (register)
0111	MOV, MOVS (register-shifted register) — rotate right
1000	TST (register)

Decode fields op	Instruction Details
1001	RSB, RSBS (immediate)
1010	CMP (register)
1011	CMN (register)
1100	ORR, ORRS (register)
1101	MUL, MULS
1110	BIC, BICS (register)
1111	MVN, MVNS (register)

Special data instructions and branch and exchange

These instructions are under 16-bit.

15 14 13 12 11 10	9 8	7 6 5 4 3 2 1 0
010001	op0	

Decode fields op0	Instruction details
11	Branch and exchange
!= 11	Add, subtract, compare, move (two high registers)

Branch and exchange

These instructions are under Special data instructions and branch and exchange.

15	14	13	12	11	10	9	8	7	6 5 4	3	2	1	0
0	1	0	0	0	1	1	1	L	Rm	(0)	(0)	(0)	

Decode fields L	Instruction Details
0	BX
1	BLX (register)

Add, subtract, compare, move (two high registers)

These instructions are under Special data instructions and branch and exchange.

15	14	13	12	11	10	9 8	7	6 5 4	3 2 1 0
0	1	0	0	0	1	!= 11 (op)	D	Rs	Rd

The following constraints also apply to this encoding: op != 11 && op != 11

op	Decode fields D:Rd	Rs	Instruction Details
00	!= 1101	!= 1101	ADD, ADDS (register)
00		1101	ADD, ADDS (SP plus register) — T1
00	1101	!= 1101	ADD, ADDS (SP plus register) — T2
01			CMP (register)
10			MOV, MOVS (register)

Load/store (register offset)

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	L	B	H		Rm			Rn			Rt	

Decode fields L	B	H	Instruction Details
0	0	0	STR (register)
0	0	1	STRH (register)
0	1	0	STRB (register)
0	1	1	LDRSB (register)
1	0	0	LDR (register)
1	0	1	LDRH (register)
1	1	0	LDRB (register)
1	1	1	LDRSH (register)

Load/store word/byte (immediate offset)

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	B	L		imm5					Rn			Rt	

Decode fields B	L	Instruction Details
0	0	STR (immediate)
0	1	LDR (immediate)
1	0	STRB (immediate)
1	1	LDRB (immediate)

Load/store halfword (immediate offset)

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	L		imm5					Rn			Rt	

Decode fields L	Instruction Details
0	STRH (immediate)
1	LDRH (immediate)

Load/store (SP-relative)

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	L		Rt			imm8						

Decode fields L	Instruction Details
0	STR (immediate)
1	LDR (immediate)

Add PC/SP (immediate)

These instructions are under [16-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	SP	Rd					imm8					

Decode fields SP	Instruction Details
0	ADR
1	ADD, ADDS (SP plus immediate)

Miscellaneous 16-bit instructions

These instructions are under [16-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1011				op0				op1	op2		op3				

op0	op1	op2	op3	Instruction details	Architecture version
0000				Adjust SP (immediate)	-
0010				Extend	-
0110	00	0		SETPAN	Armv8.1
0110	00	1		UNALLOCATED	-
0110	01			Change Processor State	-
0110	1x			UNALLOCATED	-
0111				UNALLOCATED	-
1000				UNALLOCATED	-
1010	10			HLT	-
1010	!= 10			Reverse bytes	-
1110				BKPT	-
1111			0000	Hints	-
1111			!= 0000	IT	-
x0x1				CBNZ, CBZ	-
x10x				Push and Pop	-

Adjust SP (immediate)

These instructions are under [Miscellaneous 16-bit instructions](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	0	S			imm7				

Decode fields S	Instruction Details
0	ADD, ADDS (SP plus immediate)
1	SUB, SUBS (SP minus immediate)

Extend

These instructions are under [Miscellaneous 16-bit instructions](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	1	0	U	B	Rm			Rd		

Decode fields U	B	Instruction Details
0	0	SXTH
0	1	SXTB
1	0	UXTH
1	1	UXTB

Change Processor State

These instructions are under Miscellaneous 16-bit instructions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	1	0	0	1	op	flags				

Decode fields op	flags	Instruction Details
0		SETEND
1		CPS, CPSID, CPSIE

Reverse bytes

These instructions are under Miscellaneous 16-bit instructions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	0	!= 10		Rm			Rd		
								op							

The following constraints also apply to this encoding: op != 10 && op != 10

Decode fields op	Instruction Details
00	REV
01	REV16
11	REVSH

Hints

These instructions are under Miscellaneous 16-bit instructions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	hint				0	0	0	0

Decode fields hint	Instruction Details
0000	NOP
0001	YIELD
0010	WFE
0011	WFI
0100	SEV
0101	SEVL
011x	Reserved hint, behaves as NOP
1xxx	Reserved hint, behaves as NOP

Push and Pop

These instructions are under Miscellaneous 16-bit instructions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	L	1	0	P	register_list							

Decode fields L	Instruction Details
0	PUSH
1	POP

Load/store multiple

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	L	Rn		register_list								

Decode fields L	Instruction Details
0	STM, STMIA, STMEA
1	LDM, LDMIA, LDMFD

Conditional branch, and Supervisor Call

These instructions are under 16-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1101				op0											

Decode fields op0	Instruction details
111x	Exception generation
!= 111x	B — T1

Exception generation

These instructions are under Conditional branch, and Supervisor Call.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	S	imm8							

Decode fields S	Instruction Details
0	UDF
1	SVC

32-bit

These instructions are under the top-level.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111		op0			op1							op3																			

The following constraints also apply to this encoding: op0<3:2> != 00

Decode fields op0	op1	op3	Instruction details
x11x			System register access, Advanced SIMD, and floating-point
0100	xx0xx		Load/store multiple
0100	xx1xx		Load/store dual, load/store exclusive, load-acquire/store-release, and table branch
0101			Data-processing (shifted register)
10xx		1	Branches and miscellaneous control
10x0		0	Data-processing (modified immediate)
10x1		0	Data-processing (plain binary immediate)
1100	1xxx0		Advanced SIMD element or structure load/store
1100	!= 1xxx0		Load/store single
1101	0xxxx		Data-processing (register)
1101	10xxx		Multiply, multiply accumulate, and absolute difference
1101	11xxx		Long multiply and divide

System register access, Advanced SIMD, and floating-point

These instructions are under 32-bit.

15 14 13	12	11 10	9 8	7 6 5 4 3 2 1 0	15 14 13 12	11 10 9	8 7 6 5	4	3 2 1 0
111	op0	11	op1			op2		op3	

Decode fields op0	op1	op2	op3	Instruction details
	0x	111		System register load/store and 64-bit move
	10	10x	0	Floating-point data-processing
	10	111	1	System register 32-bit move
	11			Advanced SIMD data-processing
0	0x	10x		Advanced SIMD load/store and 64-bit move
0	10	10x	1	Advanced SIMD and floating-point 32-bit move
1	0x	1x0		Advanced SIMD three registers of the same length extension
1	10	1x0		Advanced SIMD two registers and a scalar extension

System register load/store and 64-bit move

These instructions are under System register access, Advanced SIMD, and floating-point.

15 14 13	12	11 10 9	8 7 6 5	4 3 2 1 0	15 14 13 12	11 10 9	8 7 6 5 4 3 2 1 0
111		110	op0			111	

Decode fields op0	Instruction details
00x0	System register 64-bit move
!= 00x0	System register Load/Store

System register 64-bit move

These instructions are under System register load/store and 64-bit move.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15 14 13 12	11	10	9	8	7 6 5	4	3 2 1 0
1	1	1	o0	1	1	0	0	0	D	0	L	Rt2				Rt	1	1	1	cp15	opc1		CRm

| Decode fields | | | Instruction Details |
o0	D	L	
0	0		UNALLOCATED
0	1	0	MCRR
0	1	1	MRRC
1	0		UNALLOCATED
1	1		UNALLOCATED

System register Load/Store

These instructions are under [System register load/store and 64-bit move](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	o0	1	1	0	P	U	D	W	L		Rn				CRd			1	1	1	cp15				imm8				

The following constraints also apply to this encoding: P:U:D:W != 00x0

o0	P:U:W	D	L	Rn	CRd	cp15	Instruction Details
	!= 000				!= 0101	0	UNALLOCATED
	!= 000					1	UNALLOCATED
	!= 000	1			0101	0	UNALLOCATED
0	!= 000	0	1	1111	0101	0	LDC (literal)
0	0x1	0	0		0101	0	STC — post-indexed
0	0x1	0	1	!= 1111	0101	0	LDC (immediate) — post-indexed
0	010	0	0		0101	0	STC — unindexed
0	010	0	1	!= 1111	0101	0	LDC (immediate) — unindexed
0	1x0	0	0		0101	0	STC — offset
0	1x0	0	1	!= 1111	0101	0	LDC (immediate) — offset
0	1x1	0	0		0101	0	STC — pre-indexed
0	1x1	0	1	!= 1111	0101	0	LDC (immediate) — pre-indexed
1	!= 000	0			0101	0	UNALLOCATED

Floating-point data-processing

These instructions are under [System register access, Advanced SIMD, and floating-point](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	op0	1	1	1	0		op1				op2							1	0	op3			op4		0				

| Decode fields | | | | | Instruction details |
op0	op1	op2	op3	op4	
0	1x11			1	[Floating-point data-processing (two registers)](#)
0	1x11			0	[Floating-point move immediate](#)
0	!= 1x11				[Floating-point data-processing (three registers)](#)
1	0xxx		!= 00	0	[Floating-point conditional select](#)
1	1x00		!= 00		[Floating-point minNum/maxNum](#)
1	1x11	0000	!= 00	1	[Floating-point extraction and insertion](#)
1	1x11	1xxx	!= 00	1	[Floating-point directed convert to integer](#)

Floating-point data-processing (two registers)

These instructions are under [Floating-point data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	1	D	1	1	o1	opc2			Vd				1	0	size		o3	1	M	0	Vm			

o1	**Decode fields** opc2	size	o3	Instruction Details	**Architecture** **Version**
		00		UNALLOCATED	-
0	000	01	0	UNALLOCATED	-
0	000		1	VABS	-
0	000	10	0	VMOV (register) — single-precision scalar	-
0	000	11	0	VMOV (register) — double-precision scalar	-
0	001		0	VNEG	-
0	001		1	VSQRT	-
0	010		0	VCVTB — half-precision to double-precision	-
0	010	01		UNALLOCATED	-
0	010		1	VCVTT — half-precision to double-precision	-
0	011	01	0	VCVTB (BFloat16)	Armv8.6
0	011	01	1	VCVTT (BFloat16)	Armv8.6
0	011	10	0	VCVTB — single-precision to half-precision	-
0	011	10	1	VCVTT — single-precision to half-precision	-
0	011	11	0	VCVTB — double-precision to half-precision	-
0	011	11	1	VCVTT — double-precision to half-precision	-
0	100		0	VCMP — T1	-
0	100		1	VCMPE — T1	-
0	101		0	VCMP — T2	-
0	101		1	VCMPE — T2	-
0	110		0	VRINTR	-
0	110		1	VRINTZ (floating-point)	-
0	111		0	VRINTX (floating-point)	-
0	111	01	1	UNALLOCATED	-
0	111	10	1	VCVT (between double-precision and single-precision) — single-precision to double-precision	-
0	111	11	1	VCVT (between double-precision and single-precision) — double-precision to single-precision	-
1	000			VCVT (integer to floating-point, floating-point)	-
1	001	01		UNALLOCATED	-
1	001	10		UNALLOCATED	-
1	001	11	0	UNALLOCATED	-
1	001	11	1	VJCVT	Armv8.3
1	01x			VCVT (between floating-point and fixed-point, floating-point)	-
1	100		0	VCVTR	-
1	100		1	VCVT (floating-point to integer, floating-point)	-
1	101		0	VCVTR	-
1	101		1	VCVT (floating-point to integer, floating-point)	-
1	11x			VCVT (between floating-point and fixed-point, floating-point)	-

Floating-point move immediate

These instructions are under [Floating-point data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	1	D	1	1	imm4H				Vd				1	0	size		(0)	0	(0)	0	imm4L			

Decode fields size	Instruction Details	Architecture Version
00	UNALLOCATED	-
01	VMOV (immediate) — half-precision scalar	Armv8.2
10	VMOV (immediate) — single-precision scalar	-
11	VMOV (immediate) — double-precision scalar	-

Floating-point data-processing (three registers)

These instructions are under Floating-point data-processing.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	o0	D	o1			Vn					Vd			1	0		size	N	o2	M	0		Vm	

The following constraints also apply to this encoding: o0:D:o1 != 1x11

Decode fields o0:o1	size	o2	Instruction Details
!= 111	00		UNALLOCATED
000		0	VMLA (floating-point)
000		1	VMLS (floating-point)
001		0	VNMLS
001		1	VNMLA
010		0	VMUL (floating-point)
010		1	VNMUL
011		0	VADD (floating-point)
011		1	VSUB (floating-point)
100		0	VDIV
101		0	VFNMS
101		1	VFNMA
110		0	VFMA
110		1	VFMS

Floating-point conditional select

These instructions are under Floating-point data-processing.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	0	D		cc		Vn					Vd			1	0	!= 00		N	0	M	0		Vm	
																							size								

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields cc	size	Instruction Details
00		VSELEQ, VSELGE, VSELGT, VSELVS — VSELEQ
01		VSELEQ, VSELGE, VSELGT, VSELVS — VSELVS
	01	UNALLOCATED
10		VSELEQ, VSELGE, VSELGT, VSELVS — VSELGE
11		VSELEQ, VSELGE, VSELGT, VSELVS — VSELGT

Floating-point minNum/maxNum

These instructions are under [Floating-point data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	1	D	0	0		Vn				Vd			1	0	!= 00		N	op	M	0		Vm		

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields		Instruction Details
size	op	
	0	VMAXNM
01		UNALLOCATED
	1	VMINNM

Floating-point extraction and insertion

These instructions are under [Floating-point data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	1	D	1	1	0	0	0	0		Vd			1	0	!= 00		op	1	M	0		Vm		

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields		Instruction Details	Architecture Version
size	op		
01		UNALLOCATED	-
10	0	VMOVX	Armv8.2
10	1	VINS	Armv8.2
11		UNALLOCATED	-

Floating-point directed convert to integer

These instructions are under [Floating-point data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	1	D	1	1	1	o1		RM		Vd			1	0	!= 00		op	1	M	0		Vm		

size

The following constraints also apply to this encoding: size != 00 && size != 00

Decode fields			Instruction Details
o1	RM	size	
0	00		VRINTA (floating-point)
0	01		VRINTN (floating-point)
		01	UNALLOCATED
0	10		VRINTP (floating-point)
0	11		VRINTM (floating-point)
1	00		VCVTA (floating-point)
1	01		VCVTN (floating-point)
1	10		VCVTP (floating-point)
1	11		VCVTM (floating-point)

System register 32-bit move

These instructions are under [System register access, Advanced SIMD, and floating-point](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	o0	1	1	1	0		opc1		L		CRn				Rt			1	1	1	cp15		opc2		1		CRm		

Decode fields o0	L	Instruction Details
0	0	MCR
0	1	MRC
1		UNALLOCATED

Advanced SIMD data-processing

These instructions are under [System register access, Advanced SIMD, and floating-point](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111					1111			op0																			op1				

Decode fields op0	op1	Instruction details
0		[Advanced SIMD three registers of the same length](#)
1	0	[Advanced SIMD two registers, or three registers of different lengths](#)
1	1	[Advanced SIMD shifts and immediate generation](#)

Advanced SIMD three registers of the same length

These instructions are under [Advanced SIMD data-processing](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	U	1	1	1	1	0	D		size		Vn				Vd				opc			N	Q	M	o1		Vm		

	Decode fields				Instruction Details	Architecture Version
U	size	opc	Q	o1		
0	0x	1100		1	VFMA	-
0	0x	1101		0	VADD (floating-point)	-
0	0x	1101		1	VMLA (floating-point)	-
0	0x	1110		0	VCEQ (register) — T2	-
0	0x	1111		0	VMAX (floating-point)	-
0	0x	1111		1	VRECPS	-
		0000		0	VHADD	-
0	00	0001		1	VAND (register)	-
		0000		1	VQADD	-
		0001		0	VRHADD	-
0	00	1100		0	SHA1C	-
		0010		0	VHSUB	-
0	01	0001		1	VBIC (register)	-
		0010		1	VQSUB	-
		0011		0	VCGT (register) — T1	-
		0011		1	VCGE (register) — T1	-
0	01	1100		0	SHA1P	-
0	1x	1100		1	VFMS	-
0	1x	1101		0	VSUB (floating-point)	-

Decode fields					Instruction Details	Architecture Version
U	size	opc	Q	o1		
0	1x	1101		1	VMLS (floating-point)	-
0	1x	1110		0	UNALLOCATED	-
0	1x	1111		0	VMIN (floating-point)	-
0	1x	1111		1	VRSQRTS	-
		0100		0	VSHL (register)	-
0		1000		0	VADD (integer)	-
0	10	0001		1	VORR (register)	-
0		1000		1	VTST	-
		0100		1	VQSHL (register)	-
0		1001		0	VMLA (integer)	-
		0101		0	VRSHL	-
		0101		1	VQRSHL	-
0		1011		0	VQDMULH	-
0	10	1100		0	SHA1M	-
0		1011		1	VPADD (integer)	-
		0110		0	VMAX (integer)	-
0	11	0001		1	VORN (register)	-
		0110		1	VMIN (integer)	-
		0111		0	VABD (integer)	-
		0111		1	VABA	-
0	11	1100		0	SHA1SU0	-
1	0x	1101		0	VPADD (floating-point)	-
1	0x	1101		1	VMUL (floating-point)	-
1	0x	1110		0	VCGE (register) — T2	-
1	0x	1110		1	VACGE	-
1	0x	1111	0	0	VPMAX (floating-point)	-
1	0x	1111		1	VMAXNM	-
1	00	0001		1	VEOR	-
		1001		1	VMUL (integer and polynomial)	-
1	00	1100		0	SHA256H	-
		1010	0	0	VPMAX (integer)	-
1	01	0001		1	VBSL	-
		1010	0	1	VPMIN (integer)	-
		1010	1		UNALLOCATED	-
1	01	1100		0	SHA256H2	-
1	1x	1101		0	VABD (floating-point)	-
1	1x	1110		0	VCGT (register) — T2	-
1	1x	1110		1	VACGT	-
1	1x	1111	0	0	VPMIN (floating-point)	-
1	1x	1111		1	VMINNM	-
1		1000		0	VSUB (integer)	-
1	10	0001		1	VBIT	-
1		1000		1	VCEQ (register) — T1	-
1		1001		0	VMLS (integer)	-
1		1011		0	VQRDMULH	-
1	10	1100		0	SHA256SU1	-
1		1011		1	VQRDMLAH	Armv8.1

U	size	opc	Q	o1	Instruction Details	Architecture Version
	Decode fields					
1	11	0001		1	VBIF	-
1		1100		1	VQRDMLSH	Armv8.1
1		1111	1	0	UNALLOCATED	-

Advanced SIMD two registers, or three registers of different lengths

These instructions are under Advanced SIMD data-processing.

15 14 13	12	11 10 9 8 7	6	5 4	3 2 1 0	15 14 13 12	11 10 9	8 7	6 5	4	3 2 1 0
111	op0	11111		op1			op2		op3	0	

op0	op1	op2	op3	Instruction details
	Decode fields			
0	11			VEXT (byte elements)
1	11	0x		Advanced SIMD two registers misc
1	11	10		VTBL, VTBX
1	11	11		Advanced SIMD duplicate (scalar)
	!= 11		0	Advanced SIMD three registers of different lengths
	!= 11		1	Advanced SIMD two registers and a scalar

Advanced SIMD two registers misc

These instructions are under Advanced SIMD two registers, or three registers of different lengths.

15	14	13	12	11	10	9	8	7	6	5	4	3 2	1 0	15 14 13 12	11	10 9 8	7	6	5	4	3 2 1 0
1	1	1	1	1	1	1	1	1	1	D	1	1	size	opc1	Vd	0	opc2	Q	M	0	Vm

size	opc1	opc2	Q	Instruction Details	Architecture Version
	Decode fields				
	00	0000		VREV64	-
	00	0001		VREV32	-
	00	0010		VREV16	-
	00	0011		UNALLOCATED	-
	00	010x		VPADDL	-
	00	0110	0	AESE	-
	00	0110	1	AESD	-
	00	0111	0	AESMC	-
	00	0111	1	AESIMC	-
	00	1000		VCLS	-
00	10	0000		VSWP	-
	00	1001		VCLZ	-
	00	1010		VCNT	-
	00	1011		VMVN (register)	-
00	10	1100	1	UNALLOCATED	-
	00	110x		VPADAL	-
	00	1110		VQABS	-
	00	1111		VQNEG	-
	01	x000		VCGT (immediate #0)	-
	01	x001		VCGE (immediate #0)	-
	01	x010		VCEQ (immediate #0)	-

size	Decode fields opc1	opc2	Q	Instruction Details	Architecture Version
	01	x011		VCLE (immediate #0)	-
	01	x100		VCLT (immediate #0)	-
	01	x110		VABS	-
	01	x111		VNEG	-
	01	0101	1	SHA1H	-
01	10	1100	1	VCVT (from single-precision to BFloat16, Advanced SIMD)	Armv8.6
	10	0001		VTRN	-
	10	0010		VUZP	-
	10	0011		VZIP	-
	10	0100	0	VMOVN	-
	10	0100	1	VQMOVN, VQMOVUN — VQMOVUN	-
	10	0101		VQMOVN, VQMOVUN — VQMOVN	-
	10	0110	0	VSHLL	-
	10	0111	0	SHA1SU1	-
	10	0111	1	SHA256SU0	-
	10	1000		VRINTN (Advanced SIMD)	-
	10	1001		VRINTX (Advanced SIMD)	-
	10	1010		VRINTA (Advanced SIMD)	-
	10	1011		VRINTZ (Advanced SIMD)	-
10	10	1100	1	UNALLOCATED	-
	10	1100	0	VCVT (between half-precision and single-precision, Advanced SIMD) — single-precision to half-precision	-
	10	1101		VRINTM (Advanced SIMD)	-
	10	1110	0	VCVT (between half-precision and single-precision, Advanced SIMD) — half-precision to single-precision	-
	10	1110	1	UNALLOCATED	-
	10	1111		VRINTP (Advanced SIMD)	-
	11	000x		VCVTA (Advanced SIMD)	-
	11	001x		VCVTN (Advanced SIMD)	-
	11	010x		VCVTP (Advanced SIMD)	-
	11	011x		VCVTM (Advanced SIMD)	-
	11	10x0		VRECPE	-
	11	10x1		VRSQRTE	-
11	10	1100	1	UNALLOCATED	-
	11	11xx		VCVT (between floating-point and integer, Advanced SIMD)	-

Advanced SIMD duplicate (scalar)

These instructions are under Advanced SIMD two registers, or three registers of different lengths.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	D	1	1		imm4			Vd			1	1		opc		Q	M	0		Vm		

Decode fields opc	Instruction Details
000	VDUP (scalar)
001	UNALLOCATED
01x	UNALLOCATED
1xx	UNALLOCATED

Advanced SIMD three registers of different lengths

These instructions are under [Advanced SIMD two registers, or three registers of different lengths](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	U	1	1	1	1	1	1	D	!= 11						Vn				Vd				opc			N	0	M	0	Vm			

size

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields		Instruction Details
U	opc	
	0000	VADDL
	0001	VADDW
	0010	VSUBL
0	0100	VADDHN
	0011	VSUBW
0	0110	VSUBHN
0	1001	VQDMLAL
	0101	VABAL
0	1011	VQDMLSL
0	1101	VQDMULL
	0111	VABDL (integer)
	1000	VMLAL (integer)
	1010	VMLSL (integer)
1	0100	VRADDHN
1	0110	VRSUBHN
	11x0	VMULL (integer and polynomial)
1	1001	UNALLOCATED
1	1011	UNALLOCATED
1	1101	UNALLOCATED
	1111	UNALLOCATED

Advanced SIMD two registers and a scalar

These instructions are under [Advanced SIMD two registers, or three registers of different lengths](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	Q	1	1	1	1	1	1	D	!= 11						Vn				Vd				opc			N	1	M	0	Vm			

size

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields		Instruction Details	Architecture Version
Q	opc		
	000x	VMLA (by scalar)	-
0	0011	VQDMLAL	-
	0010	VMLAL (by scalar)	-
0	0111	VQDMLSL	-
	010x	VMLS (by scalar)	-
0	1011	VQDMULL	-
	0110	VMLSL (by scalar)	-
	100x	VMUL (by scalar)	-

Decode fields		Instruction Details	Architecture Version
Q	opc		
1	0011	UNALLOCATED	-
	1010	VMULL (by scalar)	-
1	0111	UNALLOCATED	-
	1100	VQDMULH	-
	1101	VQRDMULH	-
1	1011	UNALLOCATED	-
	1110	VQRDMLAH	Armv8.1
	1111	VQRDMLSH	Armv8.1

Advanced SIMD shifts and immediate generation

These instructions are under Advanced SIMD data-processing.

15 14 13	12	11 10 9 8 7	6	5 4 3 2 1 0 15 14 13 12 11 10 9 8 7 6 5	4	3 2 1 0
111		11111		op0	1	

Decode fields op0	Instruction details
000xxxxxxxxxx0	Advanced SIMD one register and modified immediate
!= 000xxxxxxxxxx0	Advanced SIMD two registers and shift amount

Advanced SIMD one register and modified immediate

These instructions are under Advanced SIMD shifts and immediate generation.

15	14	13	12	11	10	9	8	7	6	5	4	3	2 1 0	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	i	1	1	1	1	1	1	D	0	0	0 imm3	Vd	cmode	0	Q	op	1	imm4

Decode fields		Instruction Details
cmode	op	
0xx0	0	VMOV (immediate) — T1
0xx0	1	VMVN (immediate) — T1
0xx1	0	VORR (immediate) — T1
0xx1	1	VBIC (immediate) — T1
10x0	0	VMOV (immediate) — T3
10x0	1	VMVN (immediate) — T2
10x1	0	VORR (immediate) — T2
10x1	1	VBIC (immediate) — T2
11xx	0	VMOV (immediate) — T4
110x	1	VMVN (immediate) — T3
1110	1	VMOV (immediate) — T5
1111	1	UNALLOCATED

Advanced SIMD two registers and shift amount

These instructions are under Advanced SIMD shifts and immediate generation.

15	14	13	12	11	10	9	8	7	6	5 4 3	2 1 0	15 14 13 12	11 10 9 8	7	6	5	4	3 2 1 0
1	1	1	U	1	1	1	1	1	1	D imm3H	imm3L	Vd	opc	L	Q	M	1	Vm

The following constraints also apply to this encoding: imm3H:imm3L:Vd:opc:L != 000xxxxxxxxxx0

U	imm3H:L	imm3L	opc	Q	Instruction Details
	!= 0000		0000		VSHR
	!= 0000		0001		VSRA
	!= 0000	000	1010	0	VMOVL
	!= 0000		0010		VRSHR
	!= 0000		0011		VRSRA
	!= 0000		0111		VQSHL, VQSHLU (immediate) — VQSHL
	!= 0000		1001	0	VQSHRN, VQSHRUN — VQSHRN
	!= 0000		1001	1	VQRSHRN, VQRSHRUN — VQRSHRN
	!= 0000		1010	0	VSHLL
	!= 0000		11xx		VCVT (between floating-point and fixed-point, Advanced SIMD)
0	!= 0000		0101		VSHL (immediate)
0	!= 0000		1000	0	VSHRN
0	!= 0000		1000	1	VRSHRN
1	!= 0000		0100		VSRI
1	!= 0000		0101		VSLI
1	!= 0000		0110		VQSHL, VQSHLU (immediate) — VQSHLU
1	!= 0000		1000	0	VQSHRN, VQSHRUN — VQSHRUN
1	!= 0000		1000	1	VQRSHRN, VQRSHRUN — VQRSHRUN

Advanced SIMD load/store and 64-bit move

These instructions are under System register access, Advanced SIMD, and floating-point.

15 14 13 12 11 10 9	8 7 6 5	4 3 2 1 0 15 14 13 12	11 10	9 8 7 6 5 4 3 2 1 0
1110110	op0		10	

op0	Instruction details
00x0	Advanced SIMD and floating-point 64-bit move
!= 00x0	Advanced SIMD and floating-point load/store

Advanced SIMD and floating-point 64-bit move

These instructions are under Advanced SIMD load/store and 64-bit move.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	0	0	D	0	op		Rt2				Rt			1	0	size		opc2		M	o3		Vm		

D	op	size	opc2	o3	Instruction Details
0					UNALLOCATED
1				0	UNALLOCATED
1		0x	00	1	UNALLOCATED
1			01		UNALLOCATED
1	0	10	00	1	VMOV (between two general-purpose registers and two single-precision registers) — from general-purpose registers
1	0	11	00	1	VMOV (between two general-purpose registers and a doubleword floating-point register) — from general-purpose registers
1			1x		UNALLOCATED
1	1	10	00	1	VMOV (between two general-purpose registers and two single-precision registers) — to general-purpose registers

D	op	size	opc2	o3	Instruction Details
1	1	11	00	1	VMOV (between two general-purpose registers and a doubleword floating-point register) — to general-purpose registers

		Decode fields			
D	op	size	opc2	o3	Instruction Details

Advanced SIMD and floating-point load/store

These instructions are under Advanced SIMD load/store and 64-bit move.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	P	U	D	W	L		Rn				Vd			1	0	size				imm8					

The following constraints also apply to this encoding: P:U:D:W != 00x0

P	U	W	L	Rn	size	imm8	Instruction Details
0	0	1					UNALLOCATED
0	1				0x		UNALLOCATED
0	1		0		10		VSTM, VSTMDB, VSTMIA
0	1		0		11	xxxxxxx0	VSTM, VSTMDB, VSTMIA
0	1		0		11	xxxxxxx1	FSTMDBX, FSTMIAX — Increment After
0	1		1		10		VLDM, VLDMDB, VLDMIA
0	1		1		11	xxxxxxx0	VLDM, VLDMDB, VLDMIA
0	1		1		11	xxxxxxx1	FLDM*X (FLDMDBX, FLDMIAX) — Increment After
1		0	0				VSTR
1		0			00		UNALLOCATED
1		0	1	!= 1111			VLDR (immediate)
1	0	1			0x		UNALLOCATED
1	0	1	0		10		VSTM, VSTMDB, VSTMIA
1	0	1	0		11	xxxxxxx0	VSTM, VSTMDB, VSTMIA
1	0	1	0		11	xxxxxxx1	FSTMDBX, FSTMIAX — Decrement Before
1	0	1	1		10		VLDM, VLDMDB, VLDMIA
1	0	1	1		11	xxxxxxx0	VLDM, VLDMDB, VLDMIA
1	0	1	1		11	xxxxxxx1	FLDM*X (FLDMDBX, FLDMIAX) — Decrement Before
1		0	1	1111			VLDR (literal)
1	1	1					UNALLOCATED

		Decode fields					
P	U	W	L	Rn	size	imm8	Instruction Details

Advanced SIMD and floating-point 32-bit move

These instructions are under System register access, Advanced SIMD, and floating-point.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		11101110							op0												101		op1				11111				

op0	op1	Instruction details
000	0	VMOV (between general-purpose register and single-precision)
111	0	Floating-point move special register
	1	Advanced SIMD 8/16/32-bit element move/duplicate

Decode fields		Instruction details
op0	op1	

Floating-point move special register

These instructions are under Advanced SIMD and floating-point 32-bit move.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	1	1	1	L	reg				Rt				1	0	1	0	(0)	(0)	(0)	1	(0)	(0)	(0)	(0)

Decode fields L	Instruction Details
0	VMSR
1	VMRS

Advanced SIMD 8/16/32-bit element move/duplicate

These instructions are under [Advanced SIMD and floating-point 32-bit move](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	opc1		L	Vn					Rt				1	0	1	1	N	opc2		1	(0)	(0)	(0)	(0)

Decode fields opc1	L	opc2	Instruction Details
0xx	0		VMOV (general-purpose register to scalar)
	1		VMOV (scalar to general-purpose register)
1xx	0	0x	VDUP (general-purpose register)
1xx	0	1x	UNALLOCATED

Advanced SIMD three registers of the same length extension

These instructions are under [System register access, Advanced SIMD, and floating-point](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	op1		D	op2		Vn				Vd				1	op3	0	op4	N	Q	M	U	Vm			

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
x1	0x	0	0	0	0	VCADD — 64-bit SIMD vector	Armv8.3
x1	0x	0	0	0	1	UNALLOCATED	-
x1	0x	0	0	1	0	VCADD — 128-bit SIMD vector	Armv8.3
x1	0x	0	0	1	1	UNALLOCATED	-
00	0x	0	0			UNALLOCATED	-
00	0x	0	1			UNALLOCATED	-
00	00	1	0	0	0	UNALLOCATED	-
00	00	1	0	0	1	UNALLOCATED	-
00	00	1	0	1	0	VMMLA	Armv8.6
00	00	1	0	1	1	UNALLOCATED	-
00	00	1	1	0	0	VDOT (vector) — 64-bit SIMD vector	Armv8.6
00	00	1	1	0	1	UNALLOCATED	-
00	00	1	1	1	0	VDOT (vector) — 128-bit SIMD vector	Armv8.6
00	00	1	1	1	1	UNALLOCATED	-
00	01	1	0			UNALLOCATED	-
00	01	1	1			UNALLOCATED	-
00	10	0	0	0	1	VFMAL (vector) — 64-bit SIMD vector	Armv8.2
00	10	0	1			UNALLOCATED	-
00	10	1	0	0		UNALLOCATED	-
00	10	1	0	1	0	VSMMLA	Armv8.6
00	10	1	0	1	1	VUMMLA	Armv8.6
00	10	1	1	0	0	VSDOT (vector) — 64-bit SIMD vector	Armv8.2
00	10	1	1	0	1	VUDOT (vector) — 64-bit SIMD vector	Armv8.2

Top-level encodings for T32

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
		Decode fields					
00	10	1	1	1	0	VSDOT (vector) — 128-bit SIMD vector	Armv8.2
00	10	1	1	1	1	VUDOT (vector) — 128-bit SIMD vector	Armv8.2
00	11	0	0		1	VFMAB, VFMAT (BFloat16, vector)	Armv8.6
00	11	0	0	1	1	VFMAL (vector) — 128-bit SIMD vector	Armv8.2
00	11	0	1			UNALLOCATED	-
00	11	1	0			UNALLOCATED	-
00	11	1	1			UNALLOCATED	-
01	10	0	0		1	VFMSL (vector)	Armv8.2
01	10	0	1			UNALLOCATED	-
01	10	1	0	0		UNALLOCATED	-
01	10	1	0	1	0	VUSMMLA	Armv8.6
01	10	1	0	1	1	UNALLOCATED	-
01	10	1	1	0	0	VUSDOT (vector) — 64-bit SIMD vector	Armv8.6
01	10	1	1		1	UNALLOCATED	-
01	10	1	1	1	0	VUSDOT (vector) — 128-bit SIMD vector	Armv8.6
01	11					UNALLOCATED	-
	1x	0	0		0	VCMLA	Armv8.3
10	11					UNALLOCATED	-
11	11					UNALLOCATED	-

Advanced SIMD two registers and a scalar extension

These instructions are under System register access, Advanced SIMD, and floating-point.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	op1	D	op2		Vn					Vd			1	op3	0	op4	N	Q	M	U		Vm		

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
		Decode fields					
0		0	0		0	VCMLA (by element) — half-precision scalar	Armv8.3
0	00	0	0		1	VFMAL (by scalar)	Armv8.2
0	00	0	1			UNALLOCATED	-
0	00	1	0			UNALLOCATED	-
0	00	1	1	0	0	VDOT (by element) — 64-bit SIMD vector	Armv8.6
0	00	1	1		1	UNALLOCATED	-
0	00	1	1	1	0	VDOT (by element) — 128-bit SIMD vector	Armv8.6
0	01	0	0		0	UNALLOCATED	-
0	01	0	0	0	1	VFMSL (by scalar) — 64-bit SIMD vector	Armv8.2
0	01	0	0	1	1	VFMSL (by scalar) — 128-bit SIMD vector	Armv8.2
0	01	0	1			UNALLOCATED	-
0	01	1	0			UNALLOCATED	-
0	10	0				UNALLOCATED	-
0	10	1	0			UNALLOCATED	-
0	10	1	1	0	0	VSDOT (by element) — 64-bit SIMD vector	Armv8.2
0	10	1	1	0	1	VUDOT (by element) — 64-bit SIMD vector	Armv8.2
0	10	1	1	1	0	VSDOT (by element) — 128-bit SIMD vector	Armv8.2
0	10	1	1	1	1	VUDOT (by element) — 128-bit SIMD vector	Armv8.2
0	11	0	0		0	UNALLOCATED	-
0	11	0	0		1	VFMAB, VFMAT (BFloat16, by scalar)	Armv8.6

Page 90

op1	op2	op3	op4	Q	U	Instruction Details	Architecture Version
0	11	0	1			UNALLOCATED	-
0	11	1				UNALLOCATED	-
1		0	0		0	VCMLA (by element) — single-precision scalar	Armv8.3
1	00	1	1	0	0	VUSDOT (by element) — 64-bit SIMD vector	Armv8.6
1	00	1	1	0	1	VSUDOT (by element) — 64-bit SIMD vector	Armv8.6
1	00	1	1	1	0	VUSDOT (by element) — 128-bit SIMD vector	Armv8.6
1	00	1	1	1	1	VSUDOT (by element) — 128-bit SIMD vector	Armv8.6
1		0	1			UNALLOCATED	-
1	01	1	1			UNALLOCATED	-
1	1x	1	1			UNALLOCATED	-
1		1	0			UNALLOCATED	-

Decode fields header columns: op1, op2, op3, op4, Q, U; Instruction Details; **Architecture Version**

Load/store multiple

These instructions are under 32-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	opc		0	W	L	Rn				P	M	register_list													

Decode fields opc	L	Instruction Details
00	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — T1
00	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — T1
01	0	STM, STMIA, STMEA
01	1	LDM, LDMIA, LDMFD
10	0	STMDB, STMFD
10	1	LDMDB, LDMEA
11	0	SRS, SRSDA, SRSDB, SRSIA, SRSIB — T2
11	1	RFE, RFEDA, RFEDB, RFEIA, RFEIB — T2

Load/store dual, load/store exclusive, load-acquire/store-release, and table branch

These instructions are under 32-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0		op0			op1	op2											op3							

The following constraints also apply to this encoding: op0<1> == 1

op0	op1	op2	op3	Instruction details
0010				Load/store exclusive
0110	0		000	UNALLOCATED
0110	1		000	TBB, TBH
0110			01x	Load/store exclusive byte/half/dual
0110			1xx	Load-acquire / Store-release
0x11		!= 1111		Load/store dual (immediate, post-indexed)
1x10		!= 1111		Load/store dual (immediate)

Decode fields header: op0, op1, op2, op3

| 1x11 | | != 1111 | | [Load/store dual (immediate, pre-indexed)](#) |
| != 0xx0 | | 1111 | | LDRD (literal) |

Load/store exclusive

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	0	1	0	L		Rn				Rt				Rd					imm8					

Decode fields L	Instruction Details
0	STREX
1	LDREX

Load/store exclusive byte/half/dual

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	1	1	0	L		Rn				Rt				Rt2			0	1	sz		Rd			

Decode fields L	sz	Instruction Details
0	00	STREXB
0	01	STREXH
0	10	UNALLOCATED
0	11	STREXD
1	00	LDREXB
1	01	LDREXH
1	10	UNALLOCATED
1	11	LDREXD

Load-acquire / Store-release

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	1	1	0	L		Rn				Rt				Rt2			1	op	sz		Rd			

Decode fields L	op	sz	Instruction Details
0	0	00	STLB
0	0	01	STLH
0	0	10	STL
0	0	11	UNALLOCATED
0	1	00	STLEXB
0	1	01	STLEXH
0	1	10	STLEX
0	1	11	STLEXD
1	0	00	LDAB
1	0	01	LDAH
1	0	10	LDA

Decode fields			Instruction Details
L	op	sz	
1	0	11	UNALLOCATED
1	1	00	LDAEXB
1	1	01	LDAEXH
1	1	10	LDAEX
1	1	11	LDAEXD

Load/store dual (immediate, post-indexed)

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	U	1	1	L	!= 1111				Rt				Rt2				imm8							
												Rn																			

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields L	Instruction Details
0	STRD (immediate)
1	LDRD (immediate)

Load/store dual (immediate)

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	1	U	1	0	L	!= 1111				Rt				Rt2				imm8							
												Rn																			

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields L	Instruction Details
0	STRD (immediate)
1	LDRD (immediate)

Load/store dual (immediate, pre-indexed)

These instructions are under [Load/store dual, load/store exclusive, load-acquire/store-release, and table branch](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	1	U	1	1	L	!= 1111				Rt				Rt2				imm8							
												Rn																			

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields L	Instruction Details
0	STRD (immediate)
1	LDRD (immediate)

Data-processing (shifted register)

These instructions are under [32-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	1		op1			S		Rn			(0)		imm3			Rd			imm2		stype			Rm		

			Decode fields			
op1	**S**	**Rn**	**imm3:imm2:stype**	**Rd**		**Instruction Details**
0000	0					AND, ANDS (register) — AND, rotate right with extend
0000	1		!= 0000011	!= 1111		AND, ANDS (register) — ANDS, shift or rotate by value
0000	1		!= 0000011	1111		TST (register) — shift or rotate by value
0000	1		0000011	!= 1111		AND, ANDS (register) — ANDS, rotate right with extend
0000	1		0000011	1111		TST (register) — rotate right with extend
0001						BIC, BICS (register)
0010	0	!= 1111				ORR, ORRS (register) — ORR
0010	0	1111				MOV, MOVS (register) — MOV
0010	1	!= 1111				ORR, ORRS (register) — ORRS
0010	1	1111				MOV, MOVS (register) — MOVS
0011	0	!= 1111				ORN, ORNS (register) — not flag setting
0011	0	1111				MVN, MVNS (register) — MVN
0011	1	!= 1111				ORN, ORNS (register) — flag setting
0011	1	1111				MVN, MVNS (register) — MVNS
0100	0					EOR, EORS (register) — EOR, rotate right with extend
0100	1		!= 0000011	!= 1111		EOR, EORS (register) — EORS, shift or rotate by value
0100	1		!= 0000011	1111		TEQ (register) — shift or rotate by value
0100	1		0000011	!= 1111		EOR, EORS (register) — EORS, rotate right with extend
0100	1		0000011	1111		TEQ (register) — rotate right with extend
0101						UNALLOCATED
0110	0		xxxxx00			PKHBT, PKHTB — PKHBT
0110	0		xxxxx01			UNALLOCATED
0110	0		xxxxx10			PKHBT, PKHTB — PKHTB
0110	0		xxxxx11			UNALLOCATED
0111						UNALLOCATED
1000	0	!= 1101				ADD, ADDS (register) — ADD
1000	0	1101				ADD, ADDS (SP plus register) — ADD
1000	1	!= 1101		!= 1111		ADD, ADDS (register) — ADDS
1000	1	1101		!= 1111		ADD, ADDS (SP plus register) — ADDS
1000	1			1111		CMN (register)
1001						UNALLOCATED

op1	S	Rn	imm3:imm2:stype	Rd	Instruction Details
			Decode fields		
1010					ADC, ADCS (register)
1011					SBC, SBCS (register)
1100					UNALLOCATED
1101	0	!= 1101			SUB, SUBS (register) — SUB
1101	0	1101			SUB, SUBS (SP minus register) — SUB
1101	1	!= 1101		!= 1111	SUB, SUBS (register) — SUBS
1101	1	1101		!= 1111	SUB, SUBS (SP minus register) — SUBS
1101	1			1111	CMP (register)
1110					RSB, RSBS (register)
1111					UNALLOCATED

Branches and miscellaneous control

These instructions are under [32-bit](#).

15 14 13 12 11	10	9 8 7	6 5 4	3 2 1 0	15	14 13 12	11	10 9 8	7 6 5	4 3 2 1 0
11110	op0	op1	op2		1	op3		op4	op5	

op0	op1	op2	op3	op4	op5	Instruction details
		Decode fields				
0	1110	0x	0x0		0	MSR (register)
0	1110	0x	0x0		1	MSR (Banked register)
0	1110	10	0x0	000		[Hints](#)
0	1110	10	0x0	!= 000		[Change processor state](#)
0	1110	11	0x0			[Miscellaneous system](#)
0	1111	00	0x0			BXJ
0	1111	01	0x0			[Exception return](#)
0	1111	1x	0x0		0	MRS
0	1111	1x	0x0		1	MRS (Banked register)
1	1110	00	000			[DCPS](#)
1	1110	00	010			UNALLOCATED
1	1110	01	0x0			UNALLOCATED
1	1110	1x	0x0			UNALLOCATED
1	1111	0x	0x0			UNALLOCATED
1	1111	1x	0x0			[Exception generation](#)
	!= 111x		0x0			B — T3
			0x1			B — T4
			1x0			BL, BLX (immediate) — T2
			1x1			BL, BLX (immediate) — T1

Hints

These instructions are under [Branches and miscellaneous control](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7 6 5 4	3 2 1 0
1	1	1	1	0	0	1	1	1	0	1	0	(1)	(1)	(1)	(1)	1	0	(0)	0	(0)	0	0	0	hint	option

Decode fields hint	option	Instruction Details	Architecture Version
0000	0000	NOP	-
0000	0001	YIELD	-
0000	0010	WFE	-
0000	0011	WFI	-
0000	0100	SEV	-
0000	0101	SEVL	-
0000	011x	Reserved hint, behaves as NOP	-
0000	1xxx	Reserved hint, behaves as NOP	-
0001	0000	ESB	Armv8.2
0001	0001	Reserved hint, behaves as NOP	-
0001	0010	TSB CSYNC	Armv8.4
0001	0011	Reserved hint, behaves as NOP	-
0001	0100	CSDB	-
0001	0101	Reserved hint, behaves as NOP	-
0001	011x	Reserved hint, behaves as NOP	-
0001	1xxx	Reserved hint, behaves as NOP	-
001x		Reserved hint, behaves as NOP	-
01xx		Reserved hint, behaves as NOP	-
10xx		Reserved hint, behaves as NOP	-
110x		Reserved hint, behaves as NOP	-
1110		Reserved hint, behaves as NOP	-
1111		DBG	-

Change processor state

These instructions are under [Branches and miscellaneous control](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	1	0	(1)	(1)	(1)	(1)	1	0	(0)	0	(0)	imod	M	A	I	F	mode

The following constraints also apply to this encoding: imod:M != 000

Decode fields imod	M	Instruction Details
00	1	CPS, CPSID, CPSIE — CPS
01		UNALLOCATED
10		CPS, CPSID, CPSIE — CPSIE
11		CPS, CPSID, CPSIE — CPSID

Miscellaneous system

These instructions are under [Branches and miscellaneous control](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	1	1	(1)	(1)	(1)	(1)	1	0	(0)	0	(1)	(1)	(1)	(1)	opc	option

Decode fields opc	option	Instruction Details
000x		UNALLOCATED
0010		CLREX

Decode fields opc	option	Instruction Details
0011		UNALLOCATED
0100	!= 0x00	DSB
0100	0000	SSBB
0100	0100	PSSBB
0101		DMB
0110		ISB
0111		SB
1xxx		UNALLOCATED

Exception return

These instructions are under Branches and miscellaneous control.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	1	1	0	1		Rn		1	0	(0)	0	(1)	(1)	(1)	(1)				imm8				

Decode fields Rn	imm8	Instruction Details
	!= 00000000	SUB, SUBS (immediate)
1110	00000000	ERET

DCPS

These instructions are under Branches and miscellaneous control.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	1	1	1	0	0	0		imm4		1	0	0	0				imm10							opt	

Decode fields imm4	imm10	opt	Instruction Details
!= 1111			UNALLOCATED
1111	!= 0000000000		UNALLOCATED
1111	0000000000	00	UNALLOCATED
1111	0000000000	01	DCPS1
1111	0000000000	10	DCPS2
1111	0000000000	11	DCPS3

Exception generation

These instructions are under Branches and miscellaneous control.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	1	1	1	1	o1		imm4			1	0	o2	0					imm12							

Decode fields o1	o2	Instruction Details
0	0	HVC
0	1	UNALLOCATED
1	0	SMC
1	1	UDF

Data-processing (modified immediate)

These instructions are under 32-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	i	0			op1		S		Rn			0		imm3			Rd						imm8				

Decode fields				Instruction Details
op1	S	Rn	Rd	
0000	0			AND, ANDS (immediate) — AND
0000	1		!= 1111	AND, ANDS (immediate) — ANDS
0000	1		1111	TST (immediate)
0001				BIC, BICS (immediate)
0010	0	!= 1111		ORR, ORRS (immediate) — ORR
0010	0	1111		MOV, MOVS (immediate) — MOV
0010	1	!= 1111		ORR, ORRS (immediate) — ORRS
0010	1	1111		MOV, MOVS (immediate) — MOVS
0011	0	!= 1111		ORN, ORNS (immediate) — not flag setting
0011	0	1111		MVN, MVNS (immediate) — MVN
0011	1	!= 1111		ORN, ORNS (immediate) — flag setting
0011	1	1111		MVN, MVNS (immediate) — MVNS
0100	0			EOR, EORS (immediate) — EOR
0100	1		!= 1111	EOR, EORS (immediate) — EORS
0100	1		1111	TEQ (immediate)
0101				UNALLOCATED
011x				UNALLOCATED
1000	0	!= 1101		ADD, ADDS (immediate) — ADD
1000	0	1101		ADD, ADDS (SP plus immediate) — ADD
1000	1	!= 1101	!= 1111	ADD, ADDS (immediate) — ADDS
1000	1	1101	!= 1111	ADD, ADDS (SP plus immediate) — ADDS
1000	1		1111	CMN (immediate)
1001				UNALLOCATED
1010				ADC, ADCS (immediate)
1011				SBC, SBCS (immediate)
1100				UNALLOCATED
1101	0	!= 1101		SUB, SUBS (immediate) — SUB
1101	0	1101		SUB, SUBS (SP minus immediate) — SUB
1101	1	!= 1101	!= 1111	SUB, SUBS (immediate) — SUBS
1101	1	1101	!= 1111	SUB, SUBS (SP minus immediate) — SUBS
1101	1		1111	CMP (immediate)
1110				RSB, RSBS (immediate)
1111				UNALLOCATED

Data-processing (plain binary immediate)

These instructions are under 32-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	11110					1	op0		op1		0					0															

Decode fields		Instruction details
op0	op1	
0	0x	Data-processing (simple immediate)

0	10	Move Wide (16-bit immediate)
0	11	UNALLOCATED
1		Saturate, Bitfield

Data-processing (simple immediate)

These instructions are under Data-processing (plain binary immediate).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	i	1	0	o1	0	o2	0		Rn			0		imm3			Rd						imm8				

Decode fields			Instruction Details
o1	o2	Rn	
0	0	!= 11x1	ADD, ADDS (immediate)
0	0	1101	ADD, ADDS (SP plus immediate)
0	0	1111	ADR — T3
0	1		UNALLOCATED
1	0		UNALLOCATED
1	1	!= 11x1	SUB, SUBS (immediate)
1	1	1101	SUB, SUBS (SP minus immediate)
1	1	1111	ADR — T2

Move Wide (16-bit immediate)

These instructions are under Data-processing (plain binary immediate).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	i	1	0	o1	1	0	0		imm4			0		imm3			Rd						imm8				

Decode fields o1	Instruction Details
0	MOV, MOVS (immediate)
1	MOVT

Saturate, Bitfield

These instructions are under Data-processing (plain binary immediate).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	(0)	1	1		op1		0		Rn			0		imm3			Rd			imm2	(0)		widthm1				

Decode fields			Instruction Details
op1	Rn	imm3:imm2	
000			SSAT — logical shift left
001		!= 00000	SSAT — arithmetic shift right
001		00000	SSAT16
010			SBFX
011	!= 1111		BFI
011	1111		BFC
100			USAT — logical shift left
101		!= 00000	USAT — arithmetic shift right
101		00000	USAT16
110			UBFX

Decode fields			Instruction Details
op1	Rn	imm3:imm2	
111			UNALLOCATED

Advanced SIMD element or structure load/store

These instructions are under 32-bit.

15 14 13 12 11 10 9 8	7 6 5	4	3 2 1 0 15 14 13 12	11 10	9 8 7 6 5 4 3 2 1 0
11111001	op0	0		op1	

Decode fields		Instruction details
op0	op1	
0		Advanced SIMD load/store multiple structures
1	11	Advanced SIMD load single structure to all lanes
1	!= 11	Advanced SIMD load/store single structure to one lane

Advanced SIMD load/store multiple structures

These instructions are under Advanced SIMD element or structure load/store.

15	14	13	12	11	10	9	8	7	6	5	4	3 2 1 0	15 14 13 12	11 10 9 8	7 6	5 4	3 2 1 0
1	1	1	1	1	0	0	1	0	D	L	0	Rn	Vd	itype	size	align	Rm

Decode fields		Instruction Details
L	itype	
0	000x	VST4 (multiple 4-element structures)
0	0010	VST1 (multiple single elements) — T4
0	0011	VST2 (multiple 2-element structures) — T2
0	010x	VST3 (multiple 3-element structures)
0	0110	VST1 (multiple single elements) — T3
0	0111	VST1 (multiple single elements) — T1
0	100x	VST2 (multiple 2-element structures) — T1
0	1010	VST1 (multiple single elements) — T2
1	000x	VLD4 (multiple 4-element structures)
1	0010	VLD1 (multiple single elements) — T4
1	0011	VLD2 (multiple 2-element structures) — T2
1	010x	VLD3 (multiple 3-element structures)
	1011	UNALLOCATED
1	0110	VLD1 (multiple single elements) — T3
1	0111	VLD1 (multiple single elements) — T1
	11xx	UNALLOCATED
1	100x	VLD2 (multiple 2-element structures) — T1
1	1010	VLD1 (multiple single elements) — T2

Advanced SIMD load single structure to all lanes

These instructions are under Advanced SIMD element or structure load/store.

15	14	13	12	11	10	9	8	7	6	5	4	3 2 1 0	15 14 13 12	11 10	9 8	7 6	5	4	3 2 1 0
1	1	1	1	1	0	0	1	1	D	L	0	Rn	Vd	1 1	N	size	T	a	Rm

Decode fields			Instruction Details
L	N	a	
0			UNALLOCATED
1	00		VLD1 (single element to all lanes)
1	01		VLD2 (single 2-element structure to all lanes)
1	10	0	VLD3 (single 3-element structure to all lanes)
1	10	1	UNALLOCATED
1	11		VLD4 (single 4-element structure to all lanes)

Advanced SIMD load/store single structure to one lane

These instructions are under [Advanced SIMD element or structure load/store](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	1	D	L	0		Rn				Vd			!= 11		N		index_align					Rm		
																				size											

The following constraints also apply to this encoding: size != 11 && size != 11

Decode fields			Instruction Details
L	size	N	
0	00	00	VST1 (single element from one lane) — T1
0	00	01	VST2 (single 2-element structure from one lane) — T1
0	00	10	VST3 (single 3-element structure from one lane) — T1
0	00	11	VST4 (single 4-element structure from one lane) — T1
0	01	00	VST1 (single element from one lane) — T2
0	01	01	VST2 (single 2-element structure from one lane) — T2
0	01	10	VST3 (single 3-element structure from one lane) — T2
0	01	11	VST4 (single 4-element structure from one lane) — T2
0	10	00	VST1 (single element from one lane) — T3
0	10	01	VST2 (single 2-element structure from one lane) — T3
0	10	10	VST3 (single 3-element structure from one lane) — T3
0	10	11	VST4 (single 4-element structure from one lane) — T3
1	00	00	VLD1 (single element to one lane) — T1
1	00	01	VLD2 (single 2-element structure to one lane) — T1
1	00	10	VLD3 (single 3-element structure to one lane) — T1
1	00	11	VLD4 (single 4-element structure to one lane) — T1
1	01	00	VLD1 (single element to one lane) — T2
1	01	01	VLD2 (single 2-element structure to one lane) — T2
1	01	10	VLD3 (single 3-element structure to one lane) — T2
1	01	11	VLD4 (single 4-element structure to one lane) — T2
1	10	00	VLD1 (single element to one lane) — T3
1	10	01	VLD2 (single 2-element structure to one lane) — T3
1	10	10	VLD3 (single 3-element structure to one lane) — T3
1	10	11	VLD4 (single 4-element structure to one lane) — T3

Load/store single

These instructions are under [32-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1111100						op0			op1		op2									op3										

The following constraints also apply to this encoding: op0<1>:op1 != 10

op0	op1	op2	op3	Instruction details
		Decode fields		**Instruction details**
00		!= 1111	000000	Load/store, unsigned (register offset)
00		!= 1111	000001	UNALLOCATED
00		!= 1111	00001x	UNALLOCATED
00		!= 1111	0001xx	UNALLOCATED
00		!= 1111	001xxx	UNALLOCATED
00		!= 1111	01xxxx	UNALLOCATED
00		!= 1111	10x0xx	UNALLOCATED
00		!= 1111	10x1xx	Load/store, unsigned (immediate, post-indexed)
00		!= 1111	1100xx	Load/store, unsigned (negative immediate)
00		!= 1111	1110xx	Load/store, unsigned (unprivileged)
00		!= 1111	11x1xx	Load/store, unsigned (immediate, pre-indexed)
01		!= 1111		Load/store, unsigned (positive immediate)
0x		1111		Load, unsigned (literal)
10	1	!= 1111	000000	Load/store, signed (register offset)
10	1	!= 1111	000001	UNALLOCATED
10	1	!= 1111	00001x	UNALLOCATED
10	1	!= 1111	0001xx	UNALLOCATED
10	1	!= 1111	001xxx	UNALLOCATED
10	1	!= 1111	01xxxx	UNALLOCATED
10	1	!= 1111	10x0xx	UNALLOCATED
10	1	!= 1111	10x1xx	Load/store, signed (immediate, post-indexed)
10	1	!= 1111	1100xx	Load/store, signed (negative immediate)
10	1	!= 1111	1110xx	Load/store, signed (unprivileged)
10	1	!= 1111	11x1xx	Load/store, signed (immediate, pre-indexed)
11	1	!= 1111		Load/store, signed (positive immediate)
1x	1	1111		Load, signed (literal)

Load/store, unsigned (register offset)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10 9 8 7 6	5 4	3 2 1 0
1	1	1	1	1	0	0	0	0	size	L	!= 1111	Rt	0 0 0 0 0 0	imm2	Rm

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

size	L	Rt	Instruction Details
		Decode fields	**Instruction Details**
00	0		STRB (register)
00	1	!= 1111	LDRB (register)
00	1	1111	PLD, PLDW (register) — preload read
01	0		STRH (register)
01	1	!= 1111	LDRH (register)
01	1	1111	PLD, PLDW (register) — preload write
10	0		STR (register)
10	1		LDR (register)
11			UNALLOCATED

Load/store, unsigned (immediate, post-indexed)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	0	size	L	!= 1111	Rt	1 0 U 1	imm8

Rn (refers to the `!= 1111` field)

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields		Instruction Details
size	L	
00	0	STRB (immediate)
00	1	LDRB (immediate)
01	0	STRH (immediate)
01	1	LDRH (immediate)
10	0	STR (immediate)
10	1	LDR (immediate)
11		UNALLOCATED

Load/store, unsigned (negative immediate)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	0	size	L	!= 1111	Rt	1 1 0 0	imm8

Rn (refers to the `!= 1111` field)

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields			Instruction Details
size	L	Rt	
00	0		STRB (immediate)
00	1	!= 1111	LDRB (immediate)
00	1	1111	PLD, PLDW (immediate) — preload read
01	0		STRH (immediate)
01	1	!= 1111	LDRH (immediate)
01	1	1111	PLD, PLDW (immediate) — preload write
10	0		STR (immediate)
10	1		LDR (immediate)
11			UNALLOCATED

Load/store, unsigned (unprivileged)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	0	size	L	!= 1111	Rt	1 1 1 0	imm8

Rn (refers to the `!= 1111` field)

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

| Decode fields | | Instruction Details |
size	L	
00	0	STRBT
00	1	LDRBT
01	0	STRHT
01	1	LDRHT
10	0	STRT
10	1	LDRT
11		UNALLOCATED

Load/store, unsigned (immediate, pre-indexed)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10	9	8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	0	size	L	!= 1111	Rt	1 1	U	1	imm8

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

| Decode fields | | Instruction Details |
size	L	
00	0	STRB (immediate)
00	1	LDRB (immediate)
01	0	STRH (immediate)
01	1	LDRH (immediate)
10	0	STR (immediate)
10	1	LDR (immediate)
11		UNALLOCATED

Load/store, unsigned (positive immediate)
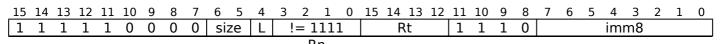
These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	1	size	L	!= 1111	Rt	imm12

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

| Decode fields | | | Instruction Details |
size	L	Rt	
00	0		STRB (immediate)
00	1	!= 1111	LDRB (immediate)
00	1	1111	PLD, PLDW (immediate) — preload read
01	0		STRH (immediate)
01	1	!= 1111	LDRH (immediate)
01	1	1111	PLD, PLDW (immediate) — preload write
10	0		STR (immediate)
10	1		LDR (immediate)

Load, unsigned (literal)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3	2	1	0	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	0	U	size	L	1	1	1	1	Rt	imm12

Decode fields			Instruction Details
size	L	Rt	
0x	1	1111	PLD (literal)
00	1	!= 1111	LDRB (literal)
01	1	!= 1111	LDRH (literal)
10	1		LDR (literal)
11			UNALLOCATED

Load/store, signed (register offset)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11	10	9	8	7	6	5	4	3 2	1 0
1	1	1	1	1	0	0	1	0	size	1	!= 1111 Rn	Rt	0	0	0	0	0	0	imm2		Rm	

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields		Instruction Details
size	Rt	
00	!= 1111	LDRSB (register)
00	1111	PLI (register)
01	!= 1111	LDRSH (register)
01	1111	Reserved hint, behaves as NOP
1x		UNALLOCATED

Load/store, signed (immediate, post-indexed)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11	10	9	8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	1	0	size	1	!= 1111 Rn	Rt	1	0	U	1	imm8

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields	Instruction Details
size	
00	LDRSB (immediate)
01	LDRSH (immediate)
1x	UNALLOCATED

Load/store, signed (negative immediate)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6 5	4	3 2 1 0	15 14 13 12	11	10	9	8	7 6 5 4 3 2 1 0
1	1	1	1	1	0	0	1	0	size	1	!= 1111 Rn	Rt	1	1	0	0	imm8

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields		Instruction Details
size	Rt	
00	!= 1111	LDRSB (immediate)
00	1111	PLI (immediate, literal)
01	!= 1111	LDRSH (immediate)
01	1111	Reserved hint, behaves as NOP
1x		UNALLOCATED

Load/store, signed (unprivileged)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	0	size		1	!= 1111				Rt				1	1	1	0	imm8							

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields	Instruction Details
size	
00	LDRSBT
01	LDRSHT
1x	UNALLOCATED

Load/store, signed (immediate, pre-indexed)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	0	size		1	!= 1111				Rt				1	1	U	1	imm8							

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields	Instruction Details
size	
00	LDRSB (immediate)
01	LDRSH (immediate)
1x	UNALLOCATED

Load/store, signed (positive immediate)

These instructions are under [Load/store single](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	1	size		1	!= 1111				Rt				imm12											

Rn

The following constraints also apply to this encoding: Rn != 1111 && Rn != 1111

Decode fields		Instruction Details
size	Rt	
00	!= 1111	LDRSB (immediate)

Decode fields		Instruction Details
size	Rt	
00	1111	PLI (immediate, literal)
01	!= 1111	LDRSH (immediate)
01	1111	Reserved hint, behaves as NOP

Load, signed (literal)

These instructions are under Load/store single.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	U	size		1	1	1	1	1	Rt				imm12											

Decode fields		Instruction Details
size	Rt	
00	!= 1111	LDRSB (literal)
00	1111	PLI (immediate, literal)
01	!= 1111	LDRSH (literal)
01	1111	Reserved hint, behaves as NOP
1x		UNALLOCATED

Data-processing (register)

These instructions are under 32-bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		11111010						op0										1111							op1							

Decode fields		Instruction details
op0	op1	
0	0000	MOV, MOVS (register-shifted register) — T2, Flag setting
0	0001	UNALLOCATED
0	001x	UNALLOCATED
0	01xx	UNALLOCATED
0	1xxx	Register extends
1	0xxx	Parallel add-subtract
1	10xx	Data-processing (two source registers)
1	11xx	UNALLOCATED

Register extends

These instructions are under Data-processing (register).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0	0	op1		U		Rn			1	1	1	1		Rd			1	(0)	rotate			Rm		

Decode fields			Instruction Details
op1	U	Rn	
00	0	!= 1111	SXTAH
00	0	1111	SXTH
00	1	!= 1111	UXTAH
00	1	1111	UXTH
01	0	!= 1111	SXTAB16
01	0	1111	SXTB16

Decode fields			Instruction Details
op1	U	Rn	
01	1	!= 1111	UXTAB16
01	1	1111	UXTB16
10	0	!= 1111	SXTAB
10	0	1111	SXTB
10	1	!= 1111	UXTAB
10	1	1111	UXTB
11			UNALLOCATED

Parallel add-subtract

These instructions are under [Data-processing (register)](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0	1		op1			Rn			1	1	1	1		Rd			0	U	H	S		Rm		

Decode fields				Instruction Details
op1	U	H	S	
000	0	0	0	SADD8
000	0	0	1	QADD8
000	0	1	0	SHADD8
000	0	1	1	UNALLOCATED
000	1	0	0	UADD8
000	1	0	1	UQADD8
000	1	1	0	UHADD8
000	1	1	1	UNALLOCATED
001	0	0	0	SADD16
001	0	0	1	QADD16
001	0	1	0	SHADD16
001	0	1	1	UNALLOCATED
001	1	0	0	UADD16
001	1	0	1	UQADD16
001	1	1	0	UHADD16
001	1	1	1	UNALLOCATED
010	0	0	0	SASX
010	0	0	1	QASX
010	0	1	0	SHASX
010	0	1	1	UNALLOCATED
010	1	0	0	UASX
010	1	0	1	UQASX
010	1	1	0	UHASX
010	1	1	1	UNALLOCATED
100	0	0	0	SSUB8
100	0	0	1	QSUB8
100	0	1	0	SHSUB8
100	0	1	1	UNALLOCATED
100	1	0	0	USUB8
100	1	0	1	UQSUB8
100	1	1	0	UHSUB8
100	1	1	1	UNALLOCATED

op1	Decode fields U	H	S	Instruction Details
101	0	0	0	SSUB16
101	0	0	1	QSUB16
101	0	1	0	SHSUB16
101	0	1	1	UNALLOCATED
101	1	0	0	USUB16
101	1	0	1	UQSUB16
101	1	1	0	UHSUB16
101	1	1	1	UNALLOCATED
110	0	0	0	SSAX
110	0	0	1	QSAX
110	0	1	0	SHSAX
110	0	1	1	UNALLOCATED
110	1	0	0	USAX
110	1	0	1	UQSAX
110	1	1	0	UHSAX
110	1	1	1	UNALLOCATED
111				UNALLOCATED

Data-processing (two source registers)

These instructions are under [Data-processing (register)](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0	1		op1			Rn			1	1	1	1			Rd			1	0		op2		Rm	

Decode fields op1	op2	Instruction Details
000	00	QADD
000	01	QDADD
000	10	QSUB
000	11	QDSUB
001	00	REV
001	01	REV16
001	10	RBIT
001	11	REVSH
010	00	SEL
010	01	UNALLOCATED
010	1x	UNALLOCATED
011	00	CLZ
011	01	UNALLOCATED
011	1x	UNALLOCATED
100	00	CRC32 — CRC32B
100	01	CRC32 — CRC32H
100	10	CRC32 — CRC32W
100	11	CONSTRAINED UNPREDICTABLE
101	00	CRC32C — CRC32CB
101	01	CRC32C — CRC32CH
101	10	CRC32C — CRC32CW
101	11	CONSTRAINED UNPREDICTABLE

Decode fields		Instruction Details
op1	op2	
11x		UNALLOCATED

The behavior of the CONSTRAINED UNPREDICTABLE encodings in this table is described in *CONSTRAINED UNPREDICTABLE behavior for A32 and T32 instruction encodings*

Multiply, multiply accumulate, and absolute difference

These instructions are under 32-bit.

15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 7	6	5 4 3 2 1 0
111110110		op0	

Decode fields	Instruction details
op0	
00	Multiply and absolute difference
01	UNALLOCATED
1x	UNALLOCATED

Multiply and absolute difference

These instructions are under Multiply, multiply accumulate, and absolute difference.

15	14	13	12	11	10	9	8	7	6 5 4	3 2 1 0 15 14 13 12	11 10 9 8	7 6 5 4	3	2	1 0
1	1	1	1	1	0	1	1	0	op1	Rn Ra	Rd	0 0	op2		Rm

Decode fields			Instruction Details
op1	Ra	op2	
000	!= 1111	00	MLA, MLAS
000		01	MLS
000		1x	UNALLOCATED
000	1111	00	MUL, MULS
001	!= 1111	00	SMLABB, SMLABT, SMLATB, SMLATT — SMLABB
001	!= 1111	01	SMLABB, SMLABT, SMLATB, SMLATT — SMLABT
001	!= 1111	10	SMLABB, SMLABT, SMLATB, SMLATT — SMLATB
001	!= 1111	11	SMLABB, SMLABT, SMLATB, SMLATT — SMLATT
001	1111	00	SMULBB, SMULBT, SMULTB, SMULTT — SMULBB
001	1111	01	SMULBB, SMULBT, SMULTB, SMULTT — SMULBT
001	1111	10	SMULBB, SMULBT, SMULTB, SMULTT — SMULTB
001	1111	11	SMULBB, SMULBT, SMULTB, SMULTT — SMULTT
010	!= 1111	00	SMLAD, SMLADX — SMLAD
010	!= 1111	01	SMLAD, SMLADX — SMLADX
010		1x	UNALLOCATED
010	1111	00	SMUAD, SMUADX — SMUAD
010	1111	01	SMUAD, SMUADX — SMUADX
011	!= 1111	00	SMLAWB, SMLAWT — SMLAWB
011	!= 1111	01	SMLAWB, SMLAWT — SMLAWT
011		1x	UNALLOCATED
011	1111	00	SMULWB, SMULWT — SMULWB
011	1111	01	SMULWB, SMULWT — SMULWT
100	!= 1111	00	SMLSD, SMLSDX — SMLSD

op1	Decode fields Ra	op2	Instruction Details
100	!= 1111	01	SMLSD, SMLSDX — SMLSDX
100		1x	UNALLOCATED
100	1111	00	SMUSD, SMUSDX — SMUSD
100	1111	01	SMUSD, SMUSDX — SMUSDX
101	!= 1111	00	SMMLA, SMMLAR — SMMLA
101	!= 1111	01	SMMLA, SMMLAR — SMMLAR
101		1x	UNALLOCATED
101	1111	00	SMMUL, SMMULR — SMMUL
101	1111	01	SMMUL, SMMULR — SMMULR
110		00	SMMLS, SMMLSR — SMMLS
110		01	SMMLS, SMMLSR — SMMLSR
110		1x	UNALLOCATED
111	!= 1111	00	USADA8
111		01	UNALLOCATED
111		1x	UNALLOCATED
111	1111	00	USAD8

Long multiply and divide

These instructions are under [32-bit](#).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	1	1		op1			Rn				RdLo				RdHi				op2				Rm		

Decode fields op1	op2	Instruction Details
000	!= 0000	UNALLOCATED
000	0000	SMULL, SMULLS
001	!= 1111	UNALLOCATED
001	1111	SDIV
010	!= 0000	UNALLOCATED
010	0000	UMULL, UMULLS
011	!= 1111	UNALLOCATED
011	1111	UDIV
100	0000	SMLAL, SMLALS
100	0001	UNALLOCATED
100	001x	UNALLOCATED
100	01xx	UNALLOCATED
100	1000	SMLALBB, SMLALBT, SMLALTB, SMLALTT — SMLALBB
100	1001	SMLALBB, SMLALBT, SMLALTB, SMLALTT — SMLALBT
100	1010	SMLALBB, SMLALBT, SMLALTB, SMLALTT — SMLALTB
100	1011	SMLALBB, SMLALBT, SMLALTB, SMLALTT — SMLALTT
100	1100	SMLALD, SMLALDX — SMLALD
100	1101	SMLALD, SMLALDX — SMLALDX
100	111x	UNALLOCATED
101	0xxx	UNALLOCATED
101	10xx	UNALLOCATED
101	1100	SMLSLD, SMLSLDX — SMLSLD
101	1101	SMLSLD, SMLSLDX — SMLSLDX

| Decode fields | | Instruction Details |
op1	op2	
101	111x	UNALLOCATED
110	0000	UMLAL, UMLALS
110	0001	UNALLOCATED
110	001x	UNALLOCATED
110	010x	UNALLOCATED
110	0110	UMAAL
110	0111	UNALLOCATED
110	1xxx	UNALLOCATED
111		UNALLOCATED

(old)	htmldiff from-	(new)