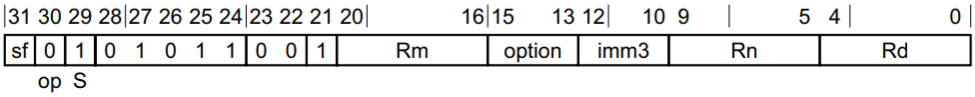
ADDS (extended register)

Add (extended register), setting flags, adds a register value and a sign or zero-extended register value, followed by an optional left shift amount, and writes the result to the destination register. The argument that is extended from the `<Rm>` register can be a byte, halfword, word, or doubleword. It updates the condition flags based on the result.

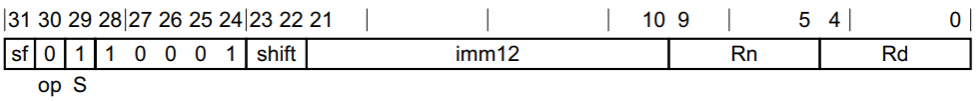
This instruction is used by the alias CMN (extended register). See *Alias conditions* for details of when each alias is preferred.



ADDS (immediate)

Add (immediate), setting flags, adds a register value and an optionally-shifted immediate value, and writes the result to the destination register. It updates the condition flags based on the result.

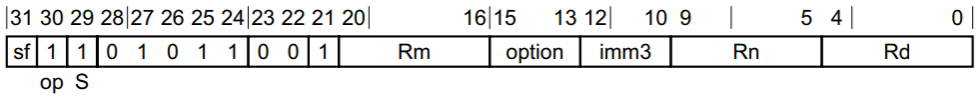
This instruction is used by the alias CMN (immediate). See *Alias conditions* for details of when each alias is preferred.



SUBS (extended register)

Subtract (extended register), setting flags, subtracts a sign or zero-extended register value, followed by an optional left shift amount, from a register value, and writes the result to the destination register. The argument that is extended from the `<Rm>` register can be a byte, halfword, word, or doubleword. It updates the condition flags based on the result.

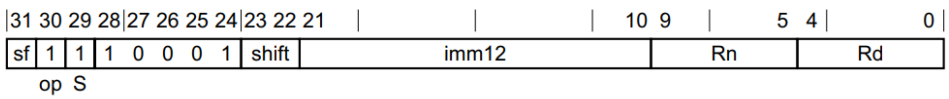
This instruction is used by the alias CMP (extended register). See *Alias conditions* for details of when each alias is preferred.



SUBS (immediate)

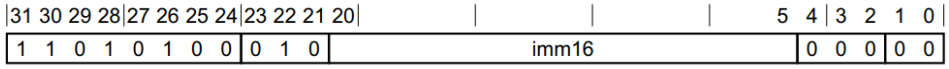
Subtract (immediate), setting flags, subtracts an optionally-shifted immediate value from a register value, and writes the result to the destination register. It updates the condition flags based on the result.

This instruction is used by the alias CMP (immediate). See *Alias conditions* for details of when each alias is preferred.



HLT

Halt instruction generates a Halt Instruction debug event.

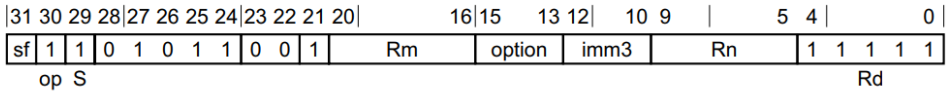


CMP (extended register)

Compare (extended register) subtracts a sign or zero-extended register value, followed by an optional left shift amount, from a register value. The argument that is extended from the `<Rm>` register can be a byte, halfword, word, or doubleword. It updates the condition flags based on the result, and discards the result.

This instruction is an alias of the SUBS (extended register) instruction. This means that:

- The encodings in this description are named to match the encodings of SUBS (extended register).
- The description of SUBS (extended register) gives the operational pseudocode for this instruction.

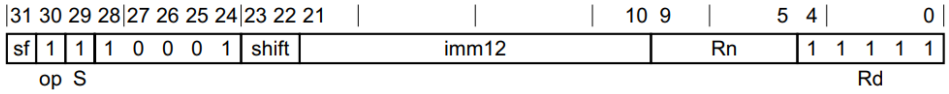


CMP (immediate)

Compare (immediate) subtracts an optionally-shifted immediate value from a register value. It updates the condition flags based on the result, and discards the result.

This instruction is an alias of the SUBS (immediate) instruction. This means that:

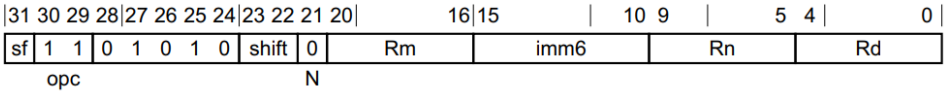
- The encodings in this description are named to match the encodings of SUBS (immediate).
- The description of SUBS (immediate) gives the operational pseudocode for this instruction.



ANDS (shifted register)

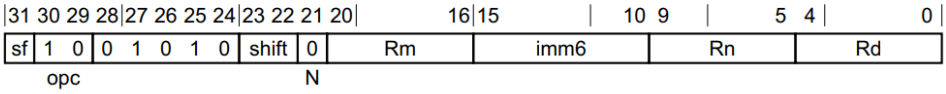
Bitwise AND (shifted register), setting flags, performs a bitwise AND of a register value and an optionally-shifted register value, and writes the result to the destination register. It updates the condition flags based on the result.

This instruction is used by the alias TST (shifted register). See *Alias conditions* for details of when each alias is preferred.



EOR (shifted register)

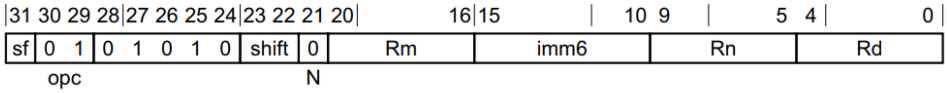
Bitwise Exclusive OR (shifted register) performs a bitwise Exclusive OR of a register value and an optionally-shifted register value, and writes the result to the destination register.



ORR (shifted register)

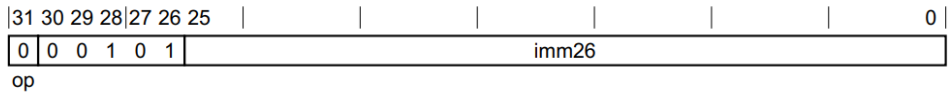
Bitwise OR (shifted register) performs a bitwise (inclusive) OR of a register value and an optionally-shifted register value, and writes the result to the destination register.

This instruction is used by the alias MOV (register). See *Alias conditions* for details of when each alias is preferred.



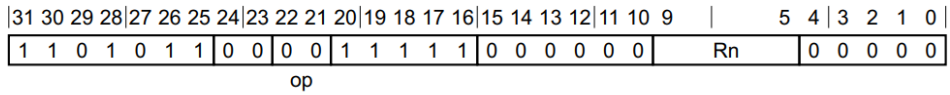
B

Branch causes an unconditional branch to a label at a PC-relative offset, with a hint that this is not a subroutine call or return.



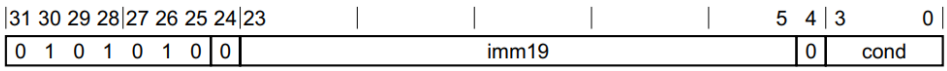
BR

Branch to Register branches unconditionally to an address in a register, with a hint that this is not a subroutine return.



B.cond

Branch conditionally to a label at a PC-relative offset, with a hint that this is not a subroutine call or return.

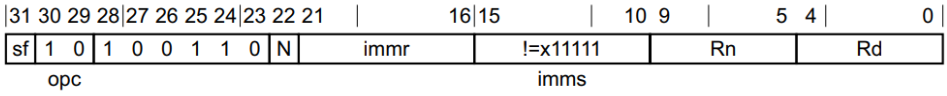


LSL (immediate)

Logical Shift Left (immediate) shifts a register value left by an immediate number of bits, shifting in zeros, and writes the result to the destination register.

This instruction is an alias of the UBFM instruction. This means that:

- The encodings in this description are named to match the encodings of UBFM.
- The description of UBFM gives the operational pseudocode for this instruction.

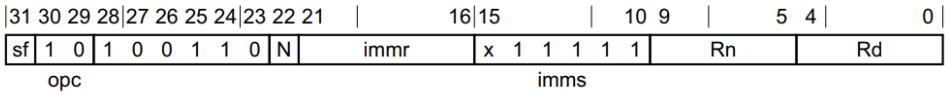


LSR (immediate)

Logical Shift Right (immediate) shifts a register value right by an immediate number of bits, shifting in zeros, and writes the result to the destination register.

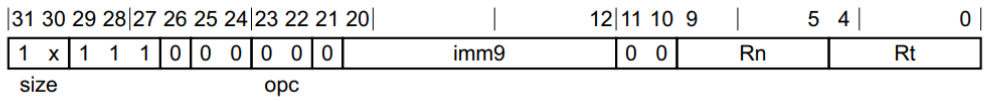
This instruction is an alias of the UBFM instruction. This means that:

- The encodings in this description are named to match the encodings of UBFM.
- The description of UBFM gives the operational pseudocode for this instruction.



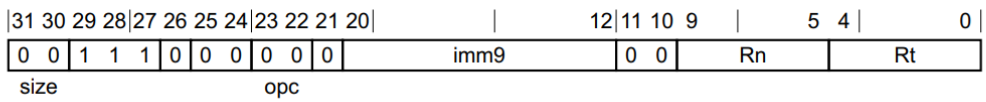
STUR

Store Register (unscaled) calculates an address from a base register value and an immediate offset, and stores a 32-bit word or a 64-bit doubleword to the calculated address, from a register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



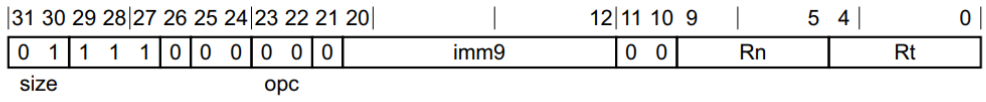
STURB

Store Register Byte (unscaled) calculates an address from a base register value and an immediate offset, and stores a byte to the calculated address, from a 32-bit register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



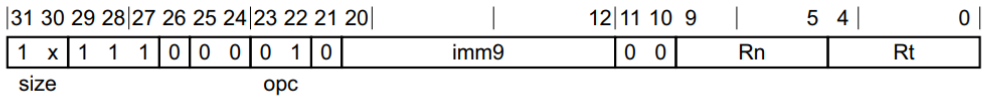
STURH

Store Register Halfword (unscaled) calculates an address from a base register value and an immediate offset, and stores a halfword to the calculated address, from a 32-bit register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



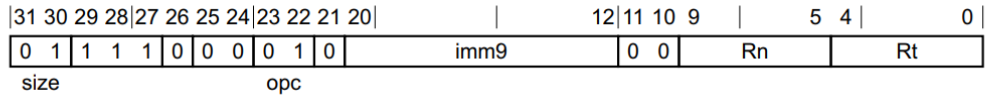
LDUR

Load Register (unscaled) calculates an address from a base register and an immediate offset, loads a 32-bit word or 64-bit doubleword from memory, zero-extends it, and writes it to a register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



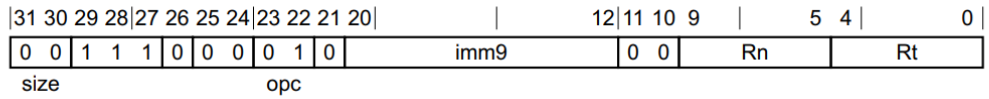
LDURH

Load Register Halfword (unscaled) calculates an address from a base register and an immediate offset, loads a halfword from memory, zero-extends it, and writes it to a register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



LDURB

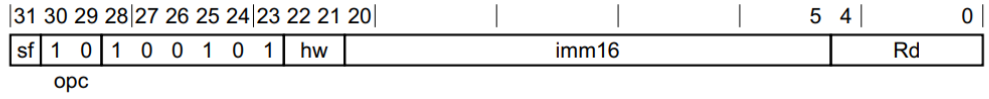
Load Register Byte (unscaled) calculates an address from a base register and an immediate offset, loads a byte from memory, zero-extends it, and writes it to a register. For information about memory accesses, see *Load/Store addressing modes* on page C1-143.



MOVZ

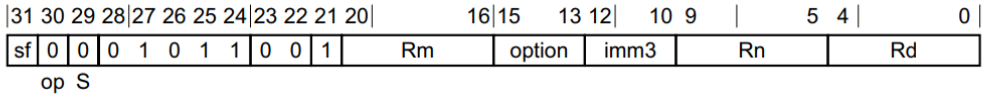
Move wide with zero moves an optionally-shifted 16-bit immediate value to a register.

This instruction is used by the alias MOV (wide immediate). See *Alias conditions* for details of when each alias is preferred.



ADD (extended register)

Add (extended register) adds a register value and a sign or zero-extended register value, followed by an optional left shift amount, and writes the result to the destination register. The argument that is extended from the <Rm> register can be a byte, halfword, word, or doubleword.



ADD (immediate)

Add (immediate) adds a register value and an optionally-shifted immediate value, and writes the result to the destination register.

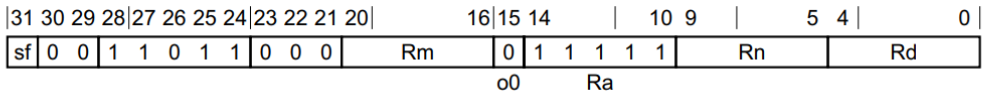
This instruction is used by the alias MOV (to/from SP). See *Alias conditions* for details of when each alias is preferred.

MUL

Multiply : `Rd = Rn * Rm`

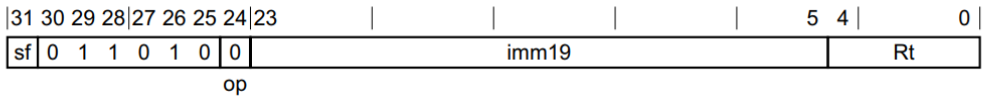
This instruction is an alias of the MADD instruction. This means that:

- The encodings in this description are named to match the encodings of MADD.
- The description of MADD gives the operational pseudocode for this instruction.



CBZ

Compare and Branch on Zero compares the value in a register with zero, and conditionally branches to a label at a PC-relative offset if the comparison is equal. It provides a hint that this is not a subroutine call or return. This instruction does not affect condition flags.



CBNZ

Compare and Branch on Nonzero compares the value in a register with zero, and conditionally branches to a label at a PC-relative offset if the comparison is not equal. It provides a hint that this is not a subroutine call or return. This instruction does not affect the condition flags.