Cortex-M3 Devices Generic User Guide

3.3.7. Conditional execution

Most data processing instructions can optionally update the condition flags in the *Application Program Status Register* (APSR) according to the result of the operation, see *Application Program Status Register*. Some instructions update all flags, and some only update a subset. If a flag is not updated, the original value is preserved. See the instruction descriptions for the flags they affect.

You can execute an instruction conditionally, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

Conditional execution is available by using conditional branches or by adding condition code suffixes to instructions. See Table 3.4 for a list of the suffixes to add to instructions to make them conditional instructions. The condition code suffix enables the processor to test a condition based on the flags. If the condition test of a conditional instruction fails, the instruction:

- does not execute
- does not write any value to its destination register
- does not affect any of the flags
- does not generate any exception.

Conditional instructions, except for conditional branches, must be inside an If-Then instruction block. See *IT* for more information and restrictions when using the `IT` instruction. Depending on the vendor, the assembler might automatically insert an `IT` instruction if you have conditional instructions outside the IT block.

Use the `CBZ` and `CBNZ` instructions to compare the value of a register against zero and branch on the result.

This section describes:

- *The condition flags*
- *Condition code suffixes*.

The condition flags

The APSR contains the following condition flags:

N

Set to 1 when the result of the operation was negative, cleared to 0 otherwise.

Z

Set to 1 when the result of the operation was zero, cleared to 0 otherwise.

C

Set to 1 when the operation resulted in a carry, cleared to 0 otherwise.

V

Set to 1 when the operation caused overflow, cleared to 0 otherwise.

For more information about the APSR see *Program Status Register*.

A carry occurs:

- if the result of an addition is greater than or equal to 2^{32}
- if the result of a subtraction is positive or zero
- as the result of an inline barrel shifter operation in a move or logical instruction.

Overflow occurs when the sign of the result, in bit[31], does not match the sign of the result had the operation been performed at infinite precision, for example:

- if adding two negative values results in a positive value
- if adding two positive values results in a negative value

- if subtracting a positive value from a negative value generates a positive value
- if subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for `CMP`, or adding, for `CMN`, except that the result is discarded. See the instruction descriptions for more information.

> **Note**
>
> Most instructions update the status flags only if the `S` suffix is specified. See the instruction descriptions for more information.

Condition code suffixes

The instructions that can be conditional have an optional condition code, shown in syntax descriptions as `{`*`cond`*`}`. Conditional execution requires a preceding `IT` instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition. Table 3.4 shows the condition codes to use.

You can use conditional execution with the `IT` instruction to reduce the number of branch instructions in code.

Table 3.4 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 3.4. Condition code suffixes

Suffix	Flags	Meaning
`EQ`	Z = 1	Equal
`NE`	Z = 0	Not equal
`CS or HS`	C = 1	Higher or same, unsigned
`CC or LO`	C = 0	Lower, unsigned
`MI`	N = 1	Negative
`PL`	N = 0	Positive or zero
`VS`	V = 1	Overflow
`VC`	V = 0	No overflow
`HI`	C = 1 and Z = 0	Higher, unsigned
`LS`	C = 0 or Z = 1	Lower or same, unsigned
`GE`	N = V	Greater than or equal, signed
`LT`	N != V	Less than, signed
`GT`	Z = 0 and N = V	Greater than, signed
`LE`	Z = 1 and N != V	Less than or equal, signed
`AL`	Can have any value	Always. This is the default when no suffix is specified.

Example 3.1 shows the use of a conditional instruction to find the absolute value of a number. `R0 = abs(R1)`.

Example 3.1. Absolute value

```
    MOVS    R0, R1          ; R0 = R1, setting flags
    IT      MI              ; skipping next instruction if value 0 or positive
    RSBMI   R0, R0, #0      ; If negative, R0 = -R0
```

Example 3.2 shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

Example 3.2. Compare and update value

```
CMP     R0, R1       ; Compare R0 and R1, setting flags
ITT     GT           ; Skip next two instructions unless GT condition holds
CMPGT   R2, R3       ; If 'greater than', compare R2 and R3, setting flags
MOVGT   R4, R5       ; If still 'greater than', do R4 = R5
```

ARM DUI 0552A

ID121610