"VLDR S0,=1.0" rejected, but accepts "VLDR S0,=1" and generates bad code.

Ref: https://bugs.launchpad.net/gcc-arm-embedded/+bug/1695572

The instruction VLDR S0,=1.0 is rejected by the assembler, but VLDR S0,=1 is accepted. However, using an integer constant does NOT load the floating-point 1.0 (0x3F800000) into S0; instead it loads the integer representation of 1 (i.e., 0x00000001) into S0.

(Has NOT been corrected as of 9-2019-q4-major of GNU ARM Embedded Toolchain.)

GNU assembler can't deal with AL condition code in IT block

Ref: https://bugs.launchpad.net/gcc-arm-embedded/+bug/1620025

As a result, the coding technique used in section 6.5.1 of the 4th edition of the text to reduce code size currently will not assemble.

(Has NOT been corrected as of 9-2019-q4-major of GNU ARM Embedded Toolchain.)

Bug #3: The pseudo-instruction LDR R8,=0 incorrectly generates CMP R0,0

(Corrected in 6-2017-q1-update of GNU ARM Embedded Toolchain)

This problem seems to happen when:

1. the destination register is one of the "high" registers (R8-R15), and
2. the constant is small (0-255).

Bug #4: The pseudo-instruction LDR R0,=1 incorrectly generates MOVS R0,1

(Corrected in 6-2017-q2-update of GNU ARM Embedded Toolchain)

The ARM documentation at http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0489i/Babbfdih.html, under the heading "LDR in Thumb code", says:

"The LDR pseudo-instruction never generates a 16-bit flag-setting MOV instruction."

However, the pseudo-instruction:

LDR R0,=1

gets translated into:

MOVS R0,1

which modifies the flags.