OperandsAndConstraints.txt

A Complete List of GCC ARM Assembly Operand Modifiers and Constraints

Valid as of Saturday, July 22, 2017.

##
######################### ASM Operand modifiers ########################
##

Valid operand modifiers for use in asm statements.

A : A memory operand for a VLD1/VST1 instruction.
a : Print as a memory address.
c : Don't print "#" before an immediate operand.
P : Print a VFP double precision register.
q : Print a NEON quad precision register.
y : Print a VFP single precision register as indexed double.
B : Bitwise inverse of integer or symbol without a preceding #.
L : The low 16 bits of an immediate constant.
M : A register range suitable for LDM/STM.
m : The base register of a memory operand.
R : The most significant register of a pair.
Q : The least significant register of a pair.
e : The low doubleword register of a NEON quad register.
f : The high doubleword register of a NEON quad register.
H : The highest-numbered register of a pair.
h : A range of VFP/NEON registers suitable for VLD1/VST1.
0-9 : %0 means "first operand, regardless of whether it's input or output," %1 means "the next operand listed after the first one," etc.

Example: Using %P[thing] in an asm instruction explicitly defines that a double is located at the memory address represented by variable "thing"

Above operands were found in LLVM's ARMAsmPrinter.cpp.

##
################## Input/Output Operand Constraint Modifiers ###########
##

These are placed in front of constraints in operands if needed (like "=r" for a write-only output general register).

"=" : Can write to the operand only.
"+" : Can read from and write to the operand.
"&" : Output operand is stored before the need for the input operands (and their index registers) is over.
"%" : This operand and the next operand are commutative (order interchangeable).

##
######################### GCC Common Constraints #######################
##

Register constraints

"r" : Matches any general register.
"g" : Any register, memory or immediate integer operand is allowed, except for registers that are not general registers.

Memory constraints

"m" : Matches any valid memory.
"o" : Matches an offsettable memory reference.

Normal constraints

"0-9" : Use this operand for the #th operand (for example: asm("mov %0, r1" : "=r" (value) : "0" (value)); means use the same (value) register that is used for input for output)
(#) Similar to +

"V" : Matches a non-offsettable memory reference.
(#) "V" matches TARGET_MEM_CONSTRAINTs that are rejected by "o".
(#) This means that it is not a memory constraint in the usual sense, since reloading the address into a base register would make the address offsettable.
(#) "o" and "V" are like components of "m" in a way.
"<" : Matches a pre-dec or post-dec operand.
">" : Matches a pre-inc or post-inc operand.
(#) Like "V", "<" and ">" are not memory constraints, since reloading the address into a base register would cause it not to match.
"i" : Matches a general integer constant.
"s" : Matches a symbolic integer constant.
"n" : Matches a non-symbolic integer constant.
"E" : Matches a floating-point constant.
"F" : Matches a floating-point constant.
(#) There is no longer a distinction between "E" and "F".
"X" : Matches anything.

Address constraints

"p" : Matches a general address.

##
###################### ARM-SPECIFIC CONSTRAINTS ########################
##

==
NOTE: Anything marked @internal should NOT BE USED in inline asm statements.
These are included purely for edification and completeness.
==

The following register constraints have been used:
- in ARM/Thumb-2 state: t, w, x, y, z
- in Thumb state: h, b
- in both states: l, c, k, q, Cs, Ts, US
In ARM state, 'l' is an alias for 'r'
'f' and 'v' were previously used for FPA and MAVERICK registers.

The following normal constraints have been used:
- in ARM/Thumb-2 state: G, I, j, J, K, L, M
- in Thumb-1 state: I, J, K, L, M, N, O
'H' was previously used for FPA.

The following multi-letter normal constraints have been used:
- in ARM/Thumb-2 state: Da, Db, Dc, Dd, Dn, Dl, DL, Do, Dv, Dy, Di, Dt, Dp, Dz
- in Thumb-1 state: Pa, Pb, Pc, Pd, Pe
- in Thumb-2 state: Pj, PJ, Ps, Pt, Pu, Pv, Pw, Px, Py
- in all states: Pf

The following memory constraints have been used:
- in ARM/Thumb-2 state: Uh, Ut, Uv, Uy, Un, Um, Us
- in ARM state: Uq
- in Thumb state: Uu, Uw
- in all states: Q

Register constraints

"t" : The VFP registers s0-s31.
"w" : The VFP registers d0-d15, or d0-d31 for VFPv3.
"x" : The VFP registers d0-d7.
"y" : The Intel iWMMX co-processor registers.
"z" : The Intel iWMMX GR registers.
"l" : In Thumb state the core registers r0-r7.
"h" : In Thumb state the core registers r8-r15.
"Ts" : For arm_restrict_it the core registers r0-r7. GENERAL_REGS otherwise.

@internal "k" : The stack register.
@internal "q" : In ARM state with LDRD support, core registers, otherwise general registers.

@internal "b" : Thumb only. The union of the low registers and the stack register.
@internal "c" : The condition code register.
@internal "Cs" : The caller save registers. Useful for sibcalls.

Normal constraints

"j" : A constant suitable for a MOVW instruction. (ARM/Thumb-2)
"I" : In ARM/Thumb-2 state a constant that can be used as an immediate value in a Data Processing instruction. In Thumb-1 state a constant in the range 0-255.
"J" : In ARM/Thumb-2 state a constant in the range -4095 to 4095. In Thumb-1 state a constant in the range -255 to -1.
"K" : In ARM/Thumb-2 state a constant that satisfies the "I" constraint if inverted. In Thumb-1 state a constant that satisfies the "I" constraint multiplied by any power of 2.
"L" : In ARM/Thumb-2 state a constant that satisfies the "I" constraint if negated. In Thumb-1 state a constant in the range -7 to 7.
"M" : In Thumb-1 state a constant that is a multiple of 4 in the range 0-1020.
"N" : Thumb-1 state a constant in the range 0-31.
"O" : In Thumb-1 state a constant that is a multiple of 4 in the range -508 to 508.
"G" : In ARM/Thumb-2 state the floating-point constant 0.
"Pf" : Memory models except relaxed, consume or release ones.

@internal The ARM state version is internal...
@internal In ARM/Thumb-2 state a constant in the range 0-32 or any power of 2.
(#) The above two lines are not typos.
@internal "Pj" : A 12-bit constant suitable for an ADDW or SUBW instruction. (Thumb-2)
@internal "PJ" : A constant that satisfies the Pj constrant if negated.
@internal "Pa" : In Thumb-1 state a constant in the range -510 to 510.
@internal "Pb" : In Thumb-1 state a constant in the range -262 to 262.
@internal "Pc" : In Thumb-1 state a constant that is in the range 1021 to 1275.
@internal "Pd" : In Thumb state a constant in the range 0 to 7.
@internal "Pe" : In Thumb-1 state a constant in the range 256 to 510.
@internal "Ps" : In Thumb-2 state a constant in the range -255 to 255.
@internal "Pt" : In Thumb-2 state a constant in the range -7 to 7.
@internal "Pu" : In Thumb-2 state a constant in the range +1 to 8.
@internal "Pv" : In Thumb-2 state a constant in the range -255 to 0.
@internal "Pw" : In Thumb-2 state a constant in the range -255 to -1.
@internal "Px" : In Thumb-2 state a constant in the range -7 to -1.
@internal "Py" : In Thumb-2 state a constant in the range 0 to 255.
@internal "Pz" : In Thumb-2 state the constant 0.
@internal "Dz" : In ARM/Thumb-2 state a vector of constant zeros.
@internal "Da" : In ARM/Thumb-2 state a const_int, const_double or const_vector that can be generated with two Data Processing insns.
@internal "Db" : In ARM/Thumb-2 state a const_int, const_double or const_vector that can be generated with three Data Processing insns.
@internal "Dc" : In ARM/Thumb-2 state a const_int, const_double or const_vector that can be generated with four Data Processing insns.
This pattern is disabled if optimizing for space or when we have load-delay slots to fill.
@internal "Dd" : In ARM/Thumb-2 state a const_int that can be used by insn adddi.
@internal "De" : In ARM/Thumb-2 state a const_int that can be used by insn anddi.
@internal "Df" : In ARM/Thumb-2 state a const_int that can be used by insn iordi.
@internal "Dg" : In ARM/Thumb-2 state a const_int that can be used by insn xordi.
@internal "Di" : In ARM/Thumb-2 state a const_int or const_double where both the high and low SImode words can be generated as immediates in 32-bit instructions.
@internal "Dn" : In ARM/Thumb-2 state a const_vector or const_int which can be loaded with a Neon vmov immediate instruction.
@internal "Dl" : In ARM/Thumb-2 state a const_vector which can be used with a Neon vorr or vbic instruction.
@internal "DL" : In ARM/Thumb-2 state a const_vector which can be used with a Neon vorn or vand instruction.
@internal "Do" : In ARM/Thumb2 state valid offset for an ldrd/strd instruction.
@internal "Dv" : In ARM/Thumb-2 state a const_double which can be used with a VFP fconsts instruction.
@internal "Dy" : In ARM/Thumb-2 state a const_double which can be used with a VFP fconstd instruction.
@internal "Dt" : In ARM/ Thumb2 a const_double which can be used with a vcvt.f32.s32 with fract bits operation.
@internal "Dp" : In ARM/ Thumb2 a const_double which can be used with a vcvt.s32.f32 with bits operation.

Memory constraints

@internal "Ua" : An address valid for loading/storing register exclusive.
@internal "Uh" : An address suitable for byte and half-word loads which does not point inside a constant pool.
@internal "Ut" : In ARM/Thumb-2 state an address valid for loading/storing opaque structure types wider than TImode.
@internal "Uv" : In ARM/Thumb-2 state a valid VFP load/store address.
@internal "Uy" : In ARM/Thumb-2 state a valid iWMMX load/store address.
@internal "Un" : In ARM/Thumb-2 state a valid address for Neon doubleword vector load/store instructions.
@internal "Um" : In ARM/Thumb-2 state a valid address for Neon element and structure load/store instructions.

@internal "Us" : In ARM/Thumb-2 state a valid address for non-offset loads/stores of quad-word values in four ARM registers.
@internal "Uq" : In ARM state an address valid in ldrsb instructions.
@internal "Q" : An address that is a single base register.
@internal "Uu" : In Thumb state an address that is valid in 16bit encoding.
@internal "Uw" : In Thumb state an address that is valid in 16bit encoding, and that can be used for unaligned accesses.
(#) The 16-bit post-increment LDR/STR accepted by thumb1_legitimate_address_p are actually LDM/STM instructions, so cannot be used to access unaligned data.
@internal "US" : US is a symbol reference.
@internal "Uz" : A memory access that is accessible as an LDC/STC operand.

We used to have constraint letters for S and R in ARM state, but all uses of these now appear to have been removed.
Additionally, we used to have a Q constraint in Thumb state, but this wasn't really a valid memory constraint. Again, all uses of this now seem to have been removed.

All GCC constraints were found in /gcc/config/arm/contraints.md and /gcc/common.md.