10.12 Instructions unique to ARM

(Original section[1])

It's hard to pick the most unusual feature of ARM, but perhaps it is the conditional execution of instructions. Every instruction starts with a 4-bit field that determines whether it will act as a nop or as a real instruction, depending on the condition codes. Hence, conditional branches are properly considered as conditionally executing the unconditional branch instruction. Conditional execution allows avoiding a branch to jump over a single instruction. It takes less code space and time to simply conditionally execute one instruction.

The 12-bit immediate field has a novel interpretation. The 8 least significant bits are zero-extended to a 32-bit value, then rotated right the number of bits specified in the first 4 bits of the field multiplied by two. Whether this split actually catches more immediates than a simple 12-bit field would be an interesting study. One advantage is that this scheme can represent all powers of two in a 32-bit word.

Operand shifting is not limited to immediates. The second register of all arithmetic and logical processing operations has the option of being shifted before being operated on. The shift options are shift left logical, shift right logical, shift right arithmetic, and rotate right. Once again, it would be interesting to see how often operations like rotate-and-add, shift -right-and-test, and so on occur in ARM programs.

Remaining instructions

Below is a list of the remaining unique instructions of the ARM architecture:

- *Block loads and stores*—Under control of a 16-bit mask within the instructions, any of the 16 registers can be loaded or stored into memory in a single instruction. These instructions can save and restore registers on procedure entry and return. These instructions can also be used for block memory copy—offering up to four times the bandwidth of a single register load-store—and today, block copies are the most important use.
- *Reverse subtract*—RSB allows the first register to be subtracted from the immediate or shifted register. RSC does the same thing, but includes the carry when calculating the difference.
- *Long multiplies*—Similarly to MIPS, Hi and Lo registers get the 64-bit signed product (SMULL) or the 64-bit unsigned product (UMULL).
- *No divide*—Like the Alpha, integer divide is not supported in hardware.
- *Conditional trap*—A common extension to the MIPS core found in desktop RISCs (COD Figure D.6.1 (Data transfer instructions not found in MIPS core ...), COD Figure D.6.2 (Arithmetic/logical instructions not found in MIPS core ...), COD Figure D.6.3 (Control instructions not found in MIPS core ...), COD Figure D.6.4 (Floating-point instructions not found in MIPS core ...)), it comes for free in the conditional execution of all ARM instructions, including SWI.
- *Coprocessor interface*—Like many of the desktop RISCs, ARM defines a full set of coprocessor instructions: data transfer, moves between general-purpose and coprocessor registers, and coprocessor operations.
- *Floating-point architecture*—Using the coprocessor interface, a floating-point architecture has been defined for ARM. It was implemented as the FPA10 coprocessor.
- *Branch and exchange instruction sets*—The BX instruction is the transition between ARM and Thumb, using the lower 31 bits of the register to set the PC and the most significant bit to determine if the mode is ARM (1) or Thumb (0).

(*1) This section is in original form.