10.7 Instructions unique to MIPS-64

(Original section[1])

MIPS has gone through five generations of instruction sets, and this evolution has generally added features found in other architectures. Here are the salient unique features of MIPS, the first several of which were found in the original instruction set.

Nonaligned data transfers

MIPS has special instructions to handle misaligned words in memory. A rare event in most programs, it is included for supporting 16-bit minicomputer applications and for doing `memcpy` and `strcpy` faster. Although most RISCs trap if you try to load a word or store a word to a misaligned address, on all architectures misaligned words can be accessed without traps by using four load byte instructions and then assembling the result using shifts and logical ORs. The MIPS load and store word left and right instructions (`LWL, LWR, SWL, SWR`) allow this to be done in just two instructions: `LWL` loads the left portion of the register and `LWR` loads the right portion of the register. `SWL` and `SWR` do the corresponding stores. The figure below shows how they work. There are also 64-bit versions of these instructions.

Figure 10.7.1: MIPS instructions for unaligned word reads (COD Figure D.7.1).

This figure assumes operation in big-endian mode. Case 1 first loads the three bytes 101, 102, and 103 into the left of R2, leaving the least significant byte undisturbed. The following `LWR` simply loads byte 104 into the least significant byte of R2, leaving the other bytes of the register unchanged using `LWL`. Case 2 first loads byte 203 into the most significant byte of R4, and the following `LWR` loads the other three bytes of R4 from memory bytes 204, 205, and 206. `LWL` reads the word with the first byte from memory, shifts to the left to discard the unneeded byte(s), and changes only those bytes in Rd. The byte(s) transferred are from the first byte to the lowest-order byte of the word. The following `LWR` addresses the last byte, right-shifts to discard the unneeded byte(s), and finally changes only those bytes of Rd. The byte(s) transferred are from the last byte up to the highest-order byte of the word. Store word left (`SWL`) is simply the inverse of `LWL`, and store word right (`SWR`) is the inverse of `LWR`. Changing to little-endian mode flips which bytes are selected and discarded. (If big-little, left-right, load-store seem confusing, don't worry; they work!)

Remaining instructions

Below is a list of the remaining unique details of the MIPS-64 architecture:

- *NOR*—This logical instruction calculates ~(Rs1 | Rs2).
- *Constant shift amount*—Nonvariable shifts use the 5-bit constant field shown in the register-register format in COD Figure D.2.3 (Instruction formats for desktop/server RISC architectures).
- *SYSCALL*—This special trap instruction is used to invoke the operating system.
- *Move to/from control registers*—`CTCi` and `CFCi` move between the integer registers and control registers.
- *Jump/call not PC-relative*—The 26-bit address of jumps and calls is not added to the PC. It is shifted left two bits and replaces the lower 28 bits of the PC. This would only make a difference if the program were located near a 256 MB boundary.
- *TLB instructions—Translation-lookaside buffer* (TLB) misses were handled in software in MIPS I, so the instruction set also had instructions for manipulating the registers of the TLB (see COD Chapter 5 (Large and Fast: Exploiting Memory Hierarchy) for more on TLBs). These registers are considered part of the "system coprocessor". Since MIPS I the instructions differ among versions of the architecture; they are more part of the implementations than part of the instruction set architecture.
- *Reciprocal and reciprocal square root*—These instructions, which do not follow IEEE 754 guidelines of proper rounding, are included apparently for applications that value speed of divide and square root more than they value accuracy.
- *Conditional procedure call instructions*—`BGEZAL` saves the return address and branches if the content of Rs1 is greater than or equal to zero, and `BLTZAL` does the same for less than zero. The purpose of these instructions is to get a PC-relative call. (There are "likely" versions of these instructions as well.)
- *Parallel single-precision floating-point operations*—As well as extending the architecture with parallel integer operations in MDMX, MIPS-64 also supports two parallel 32-bit floating-point operations on 64-bit registers in a single instruction. "Paired single" operations include add (`ADD.PS`), subtract (`SUB.PS`), compare (`C.__.PS`), convert (`CVT.PS.S`, `CVT.S.PL`, `CVT.S.PU`), negate (`NEG.PS`), absolute value (`ABS.PS`), move (`MOV.PS`, `MOVF.PS`, `MOVT.PS`), multiply (`MUL.PS`), multiply-add (`MADD.PS`), and multiply-subtract (`MSUB.PS`).

There is no specific provision in the MIPS architecture for floating-point execution to proceed in parallel with integer execution, but the MIPS implementations of floating point allow this to happen by checking to see if arithmetic interrupts are possible early in the cycle. Normally, exception detection would force serialization of execution of integer and floating-point operations.

(*1) This section is in original form.

Provide feedback on this section