10.10 Instructions unique to PowerPC

(Original section[1])

PowerPC is the result of several generations of IBM commercial RISC machines—IBM RT/PC, IBM Power1, and IBM Power2—plus the Motorola 8800.

Branch registers: Link and counter

Rather than dedicate one of the 32 general-purpose registers to save the return address on procedure call, PowerPC puts the address into a special register called the *link register*. Since many procedures will return without calling another procedure, the link doesn't always have to be saved. Making the return address a special register makes the return jump faster, since the hardware need not go through the register read pipeline stage for return jumps.

In a similar vein, PowerPC has a *count register* to be used in *for* loops where the program iterates a fixed number of times. By using a special register, the branch hardware can determine quickly whether a branch based on the count register is likely to branch, since the value of the register is known early in the execution cycle. Tests of the value of the count register in a branch instruction will automatically decrement the count register.

Given that the count register and link register are already located with the hardware that controls branches, and that one of the problems in branch prediction is getting the target address early in the pipeline, the PowerPC architects decided to make a second use of these registers. Either register can hold a target address of a conditional branch. Thus, PowerPC supplements its basic conditional branch with two instructions that get the target address from these registers (`BCLR`, `BCCTR`).

Remaining instructions

Unlike most other RISC machines, register 0 is not hardwired to the value 0. It cannot be used as a base register—that is, it generates a 0 in this case—but in base + index addressing it can be used as the index. The other unique features of the PowerPC are as follows:

- *Load multiple and store multiple* save or restore up to 32 registers in a single instruction.
- `LSW` and `STSW` permit fetching and storing of fixed- and variable-length strings that have arbitrary alignment.
- *Rotate with mask* instructions support bit field extraction and insertion. One version rotates the data and then performs logical `AND` with a mask of ones, thereby extracting a field. The other version rotates the data but only places the bits into the destination register where there is a corresponding 1 bit in the mask, thereby inserting a field.
- *Algebraic right shift* sets the carry bit (`CA`) if the operand is negative and any 1 bits are shifted out. Thus, a signed divide by any constant power of two that rounds toward 0 can be accomplished with an `SRAWI` followed by `ADDZE`, which adds `CA` to the register.
- *`CBTLZ`* will count leading zeros.
- *`SUBFIC`* computes (immediate - RA), which can be used to develop a one's or two's complement.
- *Logical shifted immediate* instructions shift the 16-bit immediate to the left 16 bits before performing AND, OR, or XOR.

(*1) This section is in original form.

Provide feedback on this section