The SuperH-3, part 7: Bitwise logical operations

devblogs.microsoft.com/oldnewthing/20190813-00

August 13, 2019

Raymond Chen

The SH-3 has a rather basic collection of bitwise logical operations.

```
AND Rm, Rn          ; Rn &= Rm
AND #imm, r0        ; r0 &= unsigned 8-bit immediate

OR  Rm, Rn          ; Rn |= Rm
OR  #imm, r0        ; r0 |= unsigned 8-bit immediate

XOR Rm, Rn          ; Rn ^= Rm
XOR #imm, r0        ; r0 ^= unsigned 8-bit immediate

NOT Rm, Rn          ; Rn = ~Rm
```

Nothing fancy. No *nor* or *nand* or *andnot* or other goofy bitwise operations. Just plain vanilla stuff. Do note that the 8-bit immediate is unsigned here.

There is also an instruction for testing bits wthout modifying anything other than the *T* flag.

```
TST Rm, Rn              ; T = ((Rn & Rm) == 0)
TST #imm, r0            ; T = ((r0 & signed 8-bit immediate) == 0)
```

The *test* instruction performs a bitwise *and* and compares the result with zero. In this case, the 8-bit immediate is signed.

But wait, there's something goofy after all: Load/modify/store instructions!

```
AND.B #imm, @(r0, GBR)  ; @(r0 + gbr) &= 8-bit immediate
OR.B  #imm, @(r0, GBR)  ; @(r0 + gbr) |= 8-bit immediate
XOR.B #imm, @(r0, GBR)  ; @(r0 + gbr) ^= 8-bit immediate
TST.B #imm, @(r0, GBR)  ; T = ((@(r0 + gbr) & 8-bit immediate) == 0)
```

These `.B` versions of the bitwise logical operations operate on a byte in memory indexed by the *r0* and *gbr* registers. Okay, so `TST.B` is not a load/modify/store; it's just a load, but I included it in this group because he wants to be with his friends.

In practice, the Microsoft compiler does not generate these instructions.

Finally, we have this guy, the only truly atomic instruction in the SH-3 instruction set.

```
TAS.B @Rn              ; T = (@Rn == 0), @Rn |= 0x80
```

The *test-and-set* instruction reads a byte from memory, compares it against zero (setting *T* accordingly), and then sets the high bit and writes the result back out. This was clearly designed for building low-level synchronization primitives, but I'm not sure anybody actually uses it.

I say that it is the only truly atomic operation because it holds the data bus locked for the duration of its operation. The load/modify/store instructions we saw above do not lock the bus, so it's possible for a coprocessor to modify the memory out from under the SH-3.

That's it for the logical operations. Next up are the bit shifting operations.

Raymond Chen

Follow