The SuperH-3, part 3: Status flags and miscellaneous instructions

devblogs.microsoft.com/oldnewthing/20190807-00

August 7, 2019

Raymond Chen

Only four of the bits in the status register are available to user-mode:

Bit	Meaning	Notes
M	Modulus	Used by division instructions
Q	Quotient	Used by division instructions
S	Saturate	Used by multiply-add instructions
T	Test	Multi-purpose flag

(There was no official meaning for the names of the registers, so I made up mnemonics for them.)

Aside from the flags used by special-purpose instructions (multiplication and division), there is basically only one flag: *T*. Each instructions decides how it wishes to consume and produce the *T* flag.

```
CLRT        ; T = 0
SETT        ; T = 1

CLRS        ; S = 0
SETS        ; S = 1
```

There are four instructions which directly set or clear two of the bits in the status register. We'll learn more about the *M* and *Q* registers when we study integer division.

```
MOVT    Rn  ; Rn = T (0 or 1)
```

There is also a special instruction to copy the *T* flag into a register. There is no converse instruction, but we'll see later how we could try to synthesize one.

Windows CE requires that the *S* flag be clear at function entry and exit.

Since there wasn't much to be said about flags, I'll use the rest of my time to cover various miscellaneous instructions.

```
MOVA @(disp, PC), r0    ; r0 = PC + disp
```

The *move address* instruction calculates the effective address of `@(disp, PC)` and stores it into *r0*. The displacement can be a multiple of 4 up to 255 × 4 = 1020.

```
SWAP.B Rm, Rn           ; Rn = Rm with bottom two bytes swapped
SWAP.W Rm, Rn           ; Rn = Rm with top and bottom words swapped
XTRCT  Rm, Rn           ; Rn = (Rn << 16) | (Rm >> 16)
```

These instructions are for byte swapping or extracting the middle 32 bits of a 64-bit value.

```
PREF  @Rn               ; prefetch memory at Rn
```

The prefetch instruction has no effect if the memory at *Rn* is inaccessible.

```
TRAPA #imm              ; trap to kernel mode
```

The `TRAPA` instruction traps to kernel mode. It carries an 8-bit unsigned immediate payload which kernel mode can use to signify anything it wishes.

```
NOP                     ; do nothing
```

Fortunately, the instruction `0000` is invalid, rather than being a nop.

```
STC     GBR, Rn         ; Rn = GBR
LDC     Rn, GBR         ; GBR = Rn
STC     PR, Rn          ; Rn = PR
LDC     Rn, PR          ; PR = Rn
```

These instructions let you move data into and out of the special registers *gbr* and *pr*. We saw *gbr* when we learned about addressing modes. We'll learn about *pr* when we get to control transfer.

Well, that wasn't very exciting yet. Let's start doing math. Next time.

Raymond Chen