The SuperH-3, part 1: Introduction

devblogs.microsoft.com/oldnewthing/20190805-00

August 5, 2019

Raymond Chen

Windows CE supported the Hitachi SuperH-3 and SuperH-4 processors. These were commonly abbreviated SH-3 and SH-4, or just SH3 and SH4, and the architecture series was known as SHx.

I'll cover the SH-3 processor in this series, with some nods to the SH-4 as they arise. But the only binaries I have available for reverse-engineering are SH-3 binaries, so that's where my focus will be.

The SH-3 is the next step in the processor series that started with the SH-1 and SH-2. It was succeeded by the SH-4 as well as the offshoots SH-3e and SH-3-DSP. The SH-4 is probably most famous for being the processor behind the Sega Dreamcast.

As with all the processor retrospective series, I'm going to focus on how Windows CE used the processor in user mode, with particular focus on the instructions you will see in compiled code.

The SH-3 is a 32-bit RISC-style (load/store) processor with fixed-length 16-bit instructions. The small instruction size permits higher code density than its contemporaries, with Hitachi claiming a code size reduction of a third to a half compared to processors with 32-bit instructions. The design was apparently so successful that ARM licensed it for their Thumb instruction set.

The SH-3 can operate in either big-endian or little-endian mode. Windows CE uses it in little-endian mode.

The SH-3 has sixteen general-purpose integer registers, each 32 bits wide, and formally named *r0* through *r15*. They are conventionally used as follows:

Register	Meaning	Preserved?
r0	return value	No
r1		No

r2		No
r3		No
r4	argument 1	No
r5	argument 2	No
r6	argument 3	No
r7	argument 4	No
r8		Yes
r9		Yes
r10		Yes
r11		Yes
r12		Yes
r13		Yes
r14, aka *fp*	frame pointer	Yes
r15, aka *sp*	stack pointer	Yes

We'll learn more about the conventions when we study calling conventions.

There are actually two sets (banks) of the first eight registers (*r0* through *r7*). User-mode code uses only bank 0, but kernel mode can choose whether it uses bank 0 or bank 1. (And when it's using one bank, kernel mode has special instructions available to access the registers from the other bank.)

The SH-3 does not support floating point operations, but the SH-4 does. There are sixteen single-precision floating point registers which are architecturally named *fpr0* through *fpr15*, but which the Microsoft assembler calls *fr0* through *fr15*. They can be paired up to produce eight double-precision floating point registers:

Double-precision register	**Single-precision register pair**	
dr0	*fr0*	*fr1*
dr2	*fr2*	*fr3*
dr4	*fr4*	*fr5*

dr6	*fr6*	*fr7*
dr8	*fr8*	*fr9*
dr10	*fr10*	*fr11*
dr12	*fr12*	*fr13*
dr14	*fr14*	*fr15*

If you try to perform a floating point operation on an SH-3, it will trap, and the kernel will emulate the instruction. As a result, floating point on an SH-3 is very slow.

Windows NT requires that the stack be kept on a 4-byte boundary. I did not observe any red zone.

There are also some special registers:

Register	Meaning	Preserved?	Notes
pc	program counter	duh	instruction pointer, must be even
gbr	global base register	No	bonus pointer register
sr	status register	No	Flags
mach	multiply and accumulate high	No	For multiply-add operations
macl	multiply and accumulate low	No	For multiply-add operations
pr	procedure register	Yes	Return address

Some calling conventions for the SH-3 say that *mach* and *macl* are preserved, or that *gbr* is reserved, but in Windows CE, they are all scratch.

We'll take a closer look at the status register later.

The architectural names for data sizes are as follows:

- **byte**: 8-bit value
- **word**: 16-bit value
- **longword**: 32-bit value
- **quadword**: 64-bit value

Unaligned memory accesses will fault. We'll look more closely at unaligned memory access later.

The SH-3 has branch delay slots. Ugh, branch delay slots. What's worse is that some branch instructions have branch delay slots and some don't. Yikes! We'll discuss this in more detail when we get to control transfer.

Instructions on the SH-3 are generally written with source on the left and destination on the right. For example,

```
    MOV     r1, r2      ; move r1 to r2
```

The SH-3 can potentially retire two instructions per cycle, although internal resource conflicts may prevent that. For example, an `ADD` can execute in parallel with a comparison instruction, but it cannot execute in parallel with a `SUB` instruction. In the case of a resource conflict, only one instruction is retired during that cycle.

After an instruction that modifies flags, the new flags are not available for a cycle, and after a load instruction, the result is not available for two cycles. There are other pipeline hazards, but those are the ones you are likely to encounter. If you try to use the results of a prior instruction too soon, the processor will stall. (Don't forget that the SH-3 is dual-issue, so two cycles can mean up to four instructions.)

Okay, that's enough background. We'll dig in next time by looking at addressing modes.

Raymond Chen

Follow