Just for fun: Which processors prefer sign-extended loads, and which prefer zero-extended loads?

devblogs.microsoft.com/oldnewthing/20230828-00

August 28, 2023

Raymond Chen

Just for fun, let’s compare what happens on different processor architectures when you load a value from memory into a larger register. Possible responses are

- Unable: The processor does not support loading data into a larger register.
- Zero-extend: The processor zero-extends the small value to the large value.
- Sign-extend: The processor sign-extends the small value to the large value.
- Either: The processor has separate instructions for sign and zero extension.
- (blank): Does not apply because the processor doesn’t have registers larger than this.

Processor	From byte	From 16-bit	From 32-bit
8086	Unable		
Alpha AXP	Unable	Unable	Sign-extend
Alpha AXP with BWX	Zero-extend	Zero-extend	Sign-extend
80386	Either	Either	
x86-64	Either	Either	Either
Intel ia64	Zero-extend	Zero-extend	Zero-extend
MIPS	Either	Either	Either
PowerPC	Zero-extend	Either	Either
SH-4	Sign-extend	Sign-extend	
ARM	Either	Either	Either
68000	Unable	Unable	
SPARC	Either	Either	Either

RISC-V	Either	Either	Either

Notes:

- The Alpha AXP with BWX is the only processor on the list where the extension mode changes between zero and sign, depending on size. This is a historical artifact: It inherited the “sign-extend” entry from the “Alpha AXP” row above it, but it turns out that bytes are usually zero-extended, so the BWX extension chose zero-extension for bytes. Words went along with bytes to minimize the weirdness.
- The x86-64 supports both zero-extended 32-to-64 loads and sign-extended 32-to-64 loads, but the zero-extended load offers a more compact encoding.

We can organize the processors by how they choose to extend small values to larger values:

Unable	8086, 68000, Alpha AXP (bytes and words)
Zero-extend	ia64, PowerPC (bytes)
Sign-extend	Alpha AXP, SH-4
Either	80386, x86-64, MIPS, ARM, SPARC, RISC-V
Mixed	Alpha AXP with BWX, PowerPC