Just for fun: What happens when you shift a register by more than the register size?

devblogs.microsoft.com/oldnewthing/20230904-00

September 4, 2023

Raymond Chen

Just for fun, let's compare what happens on different processor architectures when you shift a register by more than the register size.

Processor	Register size	Behavior
8086	Any	mod 256 (Note 1)
Alpha AXP	64	mod 64
80386	Any	mod 32
x86-64	≤ 32	mod 32
	64	mod 64
Intel ia64	64	full value
MIPS	32	mod 32
	64	mod 64
PowerPC	32	mod 64
	64	mod 128
SH-4	32	mod 32 + sign (Note 2)
ARM (Thumb-2)	Any	mod 256
ARM (AArch 64)	32	mod 32
	64	mod 64
68000	32	mod 64
SPARC	32	mod 32

RISC-V	32	mod 32
	64	mod 64

Notes:

1. On the 8086, the shift amount is given by the 8-bit `cl` register. The running time of the instruction is proportional to the number of bits shifted, and the processor does not optimize shifts that are larger than the register size, so if you ask to shift by 255 places, it will run a loop 255 times.
2. On the SH-4, the sign of the shift amount controls the direction of the shift, and the lower 5 bits control how far to shift.

I’ve added some architectures that are of historic or current interest. Do not assume that the presence of an architecture on this list implies that I will someday cover it, and don’t assume that omission of your favorite architecture means that I never will.

From the above table, you can sort of come up with a taxonomy of shifts.

	Unsigned	**Signed**
mod register size	Alpha AXP, x86-32, x86-64, MIPS, AArch64, SPARC, RISC-V	SH-4
mod 2 × register size	PowerPC, 68000	
mod 256	8086, Thumb-2	
full value	ia64	

For x86-32, I’m kind of cheating and ignoring the registers smaller than 32 bits.

Bonus chatter: The wide variety of behavior when shifting by more than the register size is one of the reasons why the C and C++ languages leave undefined what happens when you shift by more than the bit width of the shifted type.