The history of calling conventions, part 2

devblogs.microsoft.com/oldnewthing/20040107-00

January 7, 2004

Raymond Chen

Foreshadowing: This information will actually be useful in a future discussion. Well, not the fine details, but you may notice something that explains… um… it's hard to describe. Just wait for it. Curiously, it is only the 8086 and x86 platforms that have multiple calling conventions. All the others have only one! Now we're going deep into trivia that absolutely nobody remembers or even cares about: The 32-bit calling conventions you don't see any more.

All

All of the processors listed here are RISC-style, which means there are lots of registers, none of which have any particular meaning. Well, aside from the zero register which is hard-wired to zero. (It turns out zero is a very handy number to have readily available.) Any meanings attached to the registers are those imposed by the calling convention.

As a throwback to the processors of old, the "call" instruction stores the return address in a register instead of being pushed onto the stack. A good thing, too, since the processor doesn't officially know about a "stack", it being a construction of the calling convention.

As always, registers or stack space used to pass parameters may be used as scratch by the called function, as can the return value register.

You may notice that all of the RISC calling conventions are basically the same. Once again, evidence that the 8086/x86 is the weirdo. A wildly popular weirdo, mind you.

The Alpha AXP

The Alpha AXP ("AXP" being yet another of those faux-acronyms that officially doesn't stand for anything) has 32 integer registers, one of which is hard-wired to zero. By convention, one of the registers is the "stack pointer", one is the "return address" register; and two others have special meanings unrelated to parameter passing.

The first six parameters are passed in registers, with the remaining parameters on the stack. If the function is variadic, the parameters can be spilled onto the stack so they can be accessed as an array.

Seven other registers are preserved across calls, one is the return value, and the remaining thirteen are scratch. 1 zero register + 1 stack pointer + 1 return address + 2 special + 6 parameters + 7 preserved + 1 return value + 13 scratch = 32 total integer registers.

Function names on the Alpha AXP are completely undecorated.

The MIPS R4000

The first four parameters are passed in a0, a1, a2 and a3; the remainder are spilled onto the stack. What's more, there are four "dead spaces" on the stack where the four register parameters "would have been" if they had been passed on the stack. These are for use by the callee to spill the register parameters back onto the stack if desired. (Particularly handy for variadic functions.)

Function names on the MIPS are completely undecorated.

The PowerPC

The first eight parameters are passed in registers (r3 through r10), and the return address is managed manually.

I forget what happens to parameters nine and up…

Function names on the PowerPC are decorated by prepending two periods.

Postclaimer: I haven't had personal experience with the MIPS or PPC processors, so my discussion of those processors may be a tad off, but the basic idea I think is sound.

Raymond Chen

Follow