WIKIPEDIA

x86 calling conventions

This article describes the **calling conventions** used when programming **x86** architecture microprocessors.

Calling conventions describe the interface of called code:

- The order in which atomic (scalar) parameters, or individual parts of a complex parameter, are allocated
- How parameters are passed (pushed on the stack, placed in registers, or a mix of both)
- Which registers the called function must preserve for the caller (also known as: callee-saved registers or non-volatile registers)
- How the task of preparing the stack for, and restoring after, a function call is divided between the caller and the callee

This is intimately related with the assignment of sizes and formats to programming-language types. Another closely related topic is name mangling, which determines how symbol names in the code are mapped to symbol names used by the linker. Calling conventions, type representations, and name mangling are all part of what is known as an application binary interface (ABI).

There are subtle differences in how various compilers implement these conventions, so it is often difficult to interface code which is compiled by different compilers. On the other hand, conventions which are used as an API standard (such as stdcall) are very uniformly implemented.

Contents

Historical background

Prior to microcomputers, the machine manufacturer generally provided an operating system and compilers for several programming languages. The calling convention(s) for each platform were those defined by the manufacturer's programming tools.

Early microcomputers before the Commodore Pet and Apple II generally came without an OS or compilers. The IBM PC came with Microsoft's fore-runner to Windows, the Disk Operating System (DOS), but it did not come with a compiler. The only hardware standard for IBM PC-compatible machines was defined by the Intel processors (8086, 80386) and the literal hardware IBM shipped. Hardware extensions and all software standards (save for a BIOS calling convention) were thrown open to market competition.

A multitude of independent software firms offered operating systems, compilers for many programming languages, and applications. Many different calling schemes were implemented by the firms, often mutually exclusive, based on different requirements, historical practices, and programmer creativity.

After the IBM-compatible market shakeout, Microsoft operating systems and programming tools (with differing conventions) predominated, while second-tier firms like Borland and Novell, and open-source projects like GCC, still maintained their own standards. Provisions for interoperability between vendors and products were eventually adopted, simplifying the problem of choosing a viable convention.[1]

Caller clean-up

In these conventions, the caller cleans the arguments from the stack.

cdecl