x86-64 Assembly

Architecture, Opcodes, Primitives, Functions, Flow Control, OS Calls, C interop, Virt, SIMD, Project

"What Andy giveth, Bill taketh away". As the processor folks manage to speed up the computer, the software folks invariably manage to slow it down with layer upon layer of software inserted between the application and the CPU. It is a major problem that many of today's application developers have little or no appreciation of what is happening "under the hood" as their code mysteriously interacts through all these opaque layers with the CPU. The solution is for developers to learn assembly for the CPU their applications use. This intensive course provides a whirlwind tour of major features of 64-bit x86 assembly.

There are three real benefits to learning x86-64 assembly. Firstly, you can program the performance-critical sections of your app directly in it, and exploit rich features such as SIMD. Secondly, you can optimize code written in high-level languages (HLL). With more knowledge, you can make more informed decisions about structuring your HLL code. Thirdly, you can debug complex problems at the instruction set level. The x86 opcodes are what gets passed to the CPU to execute. In tough debugging scenarios the more you understand what is transpiring, the quicker you catch bugs.

Target Audience

This course target experienced software developers who wish to learn about x86-64 assembly, so that they can code in it directly and so that they understand how their high-level languages actually execute, to help with debugging and optimization.

Prerequisites

Good experience of software programming.

No previous assembly programming experience required.

Contents of One-Day Training Course

Assembly Features

Common aspects of assembly
Assembly as a language
Opcodes
CPU Instruction Set Architecture (ISA)

Microprocessor Architecture

Machine code, microcode & micro-ops
The CPU's role in a computer system
The bus, devices, I/O, addressing

The x86 Instruction Set

What makes up the instruction set?
Registers, cache, SIMD, float-point, etc.
Traps and interrupts
Application structure

Tools

Assembler, disassembler, profiler, debugger, dumper, other

Integers And Floats

Representing scalar and floating numbers
Basic mathematical calculations

Strings

String operations
Storing, loading and manipulating

Conditionals

Flow control opcodes
Labels, jumps, branches

Functions

Calling functions
Creating functions
Parameters

Executable Formats

ELF (Linux) and PE/COFF (Windows)
How assembly is placed inside executable

The OS and System Calls

How system calls are exposed to assembly
Calling the OS API
Error handling

C and Assembly

Passing data from one to the other
How C code is accessible from assembly
How assembly is accessible from C
Inline assembly

Optimization

Understanding the cost of each instruction
Reducing the number of instructions
Using more appropriate instructions

Stream Computing

Vector programming
Single instruction, multiple data (SIMD)
Opcodes for SIMD
Highly efficient for certain workloads

Virtualization

Hypervisor
Doing virtualization in software
Modern on-chip virtualization
Working with virtualization in assembly

64-bit

Lengthening the registers
Address space changes

Other Microprocessor Families

Contrast x86 with competitors (e.g. PowerPC, ARM)
Interesting futuristic architectures

Project

Developing a high-performance custom solution in x86-64 assembly