

# 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.

<b>Contents of One-Day Training Course</b>	
<p><b>Target Audience</b> 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.</p> <p><b>Prerequisites</b> Good experience of software programming.  No previous assembly programming experience required.</p>	<div style="display: flex; justify-content: space-between;"> <div style="width: 48%;"> <p style="text-align: center;"><b>Assembly Features</b></p> <p>Common aspects of assembly Assembly as a language Opcodes CPU Instruction Set Architecture (ISA)</p> <p style="text-align: center;"><b>Microprocessor Architecture</b></p> <p>Machine code, microcode &amp; micro-ops The CPU’s role in a computer system The bus, devices, I/O, addressing</p> <p style="text-align: center;"><b>The x86 Instruction Set</b></p> <p>What makes up the instruction set? Registers, cache, SIMD, float-point, etc. Traps and interrupts Application structure</p> <p style="text-align: center;"><b>Tools</b></p> <p>Assembler, disassembler, profiler, debugger, dumper, other</p> <p style="text-align: center;"><b>Integers And Floats</b></p> <p>Representing scalar and floating numbers Basic mathematical calculations</p> <p style="text-align: center;"><b>Strings</b></p> <p>String operations Storing, loading and manipulating</p> <p style="text-align: center;"><b>Conditionals</b></p> <p>Flow control opcodes Labels, jumps, branches</p> <p style="text-align: center;"><b>Functions</b></p> <p>Calling functions Creating functions Parameters</p> <p style="text-align: center;"><b>Executable Formats</b></p> <p>ELF (Linux) and PE/COFF (Windows) How assembly is placed inside executable</p> </div> <div style="width: 48%;"> <p style="text-align: center;"><b>The OS and System Calls</b></p> <p>How system calls are exposed to assembly Calling the OS API Error handling</p> <p style="text-align: center;"><b>C and Assembly</b></p> <p>Passing data from one to the other How C code is accessible from assembly How assembly is accessible from C Inline assembly</p> <p style="text-align: center;"><b>Optimization</b></p> <p>Understanding the cost of each instruction Reducing the number of instructions Using more appropriate instructions</p> <p style="text-align: center;"><b>Stream Computing</b></p> <p>Vector programming Single instruction, multiple data (SIMD) Opcodes for SIMD Highly efficient for certain workloads</p> <p style="text-align: center;"><b>Virtualization</b></p> <p>Hypervisor Doing virtualization in software Modern on-chip virtualization Working with virtualization in assembly</p> <p style="text-align: center;"><b>64-bit</b></p> <p>Lengthening the registers Address space changes</p> <p style="text-align: center;"><b>Other Microprocessor Families</b></p> <p>Contrast x86 with competitors (e.g. PowerPC, ARM) Interesting futuristic architectures</p> <p style="text-align: center;"><b>Project</b></p> <p>Developing a high-performance custom solution in x86-64 assembly</p> </div> </div>