


## Course Description

Facilitator Information	
Name	Assoc. Prof. Goh Wooi Boon
Biodata	<p>Associate Professor Goh Wooi Boon lectures at the School of Computer Engineering (SCE) at Nanyang Technological University. He has been instructing both undergraduates and graduates in the area of microprocessors and assembly language programming for more than 12 years. He was awarded Teacher-of-the-year in 2000 and has designed challenging practical-oriented programmes for the undergraduates at SCE. Prior to joining NTU, he was involved in the design and development of automated systems and robotic workcells at Hewlett Packard Singapore Pte. Ltd. He has conducted industry-oriented courses for IT professionals and engineers from IDA, DSO, MHA, HP, AMD, etc.</p> 
Organisation	Nanyang Technological University
Email Address	<a href="mailto:aswbgoh@ntu.edu.sg">aswbgoh@ntu.edu.sg</a>
Course Information	
Title	Reverse Code Engineering – A Practical Analysis of 80x86 Assembly Code
Description	Reverse code engineering is often needed when the source code of an important program is unavailable but its functionalities have to be understood, duplicated or extended. This course will allow you to practise using software tools that will aid in this task. You will also learn how programs written in high-level languages such as C/C++ translate down to low-level 80x86 assembly codes. Relevant issues relating to software vulnerabilities and security such as stack/buffer overflow, password cracking, code optimization, and efficient arithmetic implementations will be covered. This practical-oriented course will provide you with challenging and fun exercises such as overflowing the stack, cracking a 4-digit random number generator and many more. The practical nature of these exercises is designed to help you understand difficult concepts and provide working professional with hands-on skills that can be immediately useful at the work place. Freeware tools will be provided to all participants both for the course and for their subsequent use in future reverse code engineering problems.
Objectives	<ol style="list-style-type: none"> <li>1. Understand the basic functionalities and uses of reverse code engineering tools such as disassemblers and debuggers.</li> <li>2. Understand the low-level implementation of various program and data constructs through analysis of simple 80x86 assembly code.</li> <li>3. Practise basic reverse code engineering techniques on Win32 programs.</li> </ol>
Programme	<p><b>Tools of the Trade</b></p> <ul style="list-style-type: none"> <li>- Disassembler and debugger</li> </ul> <p><b>Brief Review</b></p> <ul style="list-style-type: none"> <li>- Basic program structure</li> <li>- Data and number representation</li> <li>- 80x86 instruction set and format</li> <li>- 80x86 addressing modes</li> </ul> <p><b>Understanding Functions</b></p> <ul style="list-style-type: none"> <li>- Function call and return</li> <li>- Calling conventions (cdecl, stdcall, fastcall, etc)</li> <li>- Passing by reference and by value</li> </ul> <p><b>Understanding Basic Data Constructs</b></p> <ul style="list-style-type: none"> <li>- Global and static variables</li> </ul>

	<ul style="list-style-type: none"> <li>- Local and temporary variables</li> <li>- Strings, structures, classes</li> <li>- Buffer and stack overflow</li> </ul> <p><b>Understanding Basic Code Constructs</b></p> <ul style="list-style-type: none"> <li>- Conditional codes (IF-ELSE), Branchless logic</li> <li>- SWITCH statements</li> <li>- Loops (WHILE, DO, FOR)</li> </ul> <p><b>Compiler Optimization</b></p> <ul style="list-style-type: none"> <li>- Speed and Size optimization</li> </ul> <p><b>Understanding Compiled Integer Arithmetic</b></p> <ul style="list-style-type: none"> <li>- Addition, subtraction</li> <li>- Division, multiplication, modulo</li> <li>- Efficient implementation of division and multiplication</li> </ul> <p><b>Reverse Code Engineering Exercises and Challenges</b></p> <ul style="list-style-type: none"> <li>- Analysis of some simple Win32 programs</li> <li>- Overflowing the stack</li> <li>- Cracking simple string-based passwords</li> <li>- Enabling cripple-ware</li> <li>- Predicting a 4D number generator</li> <li>- Unraveling a complex conditional construct</li> </ul>
Targeted audience	<p>This course is most suitable for IT professionals or computer engineers whose job involves:</p> <ul style="list-style-type: none"> <li>- Analysis of Windows-based codes for security threats and quality assurance.</li> <li>- Maintenance and analysis of legacy binaries written in 80x86.</li> <li>- Writing efficient code in low-level (x86) and high-level languages (C/C++).</li> <li>- Analysing 80x86 assembly code.</li> </ul>
Pre-requisite for the course	<p>To get the most out of this course, participants should have some prior exposure to 80x86 assembly language programming. A very short review of 80x86 assembly language programming and instruction set will be done at the beginning of the course as a brief refresher.</p> <p>Participants must bring along their own Notebooks (with Windows XP operating system installed).</p>
Instructional strategies	<p>This is a practical-oriented course involving hands-on exercises after each short topical study.</p> <p>Freeware tools will be provided for participants to do both static and live code analysis.</p>
<b>Course Schedule and Costing</b>	
Other Remarks	<p><b>Important:</b></p> <p>Because of the practical nature of the course, all participants must bring along their own <b>Notebook</b> with the <b>Windows XP operating system</b> installed.</p> <p>The necessary software that will be used during the course will be provided for you before the commencement of the course.</p>