Assembly_Language

Course Content Outline
I. Basic Concepts of Assembly Language
A. Why learn assembly language
B. How data are represented
C. Boolean expressions
1. What is binary logic
2. How to use truth tables
3. Boolean functions

II. x86 Processor Architecture
A. How the instruction execution cycle works
B. The Modes of operation and how they are used
C. How registers and flags manage memory
D. How the floating point unit performs mathematical functions

III. Fundamentals of Assembly Language
A. How to declare literals and constants
1. What are reserved words
2. When to use identifiers
3. What are directives
B. The 4 parts of an instruction
1. Label
2. Instruction mnemonic
3. Operands
4. Comment
C. What are intrinsic data type definitions
D. The x86 uses Little-Endian vs. Big-Endian order
E. Using symbolic constants

IV. Basic Programming Syntax
A. Performing data transfers
B. Performing simple arithmetic
C. Performing data related operations
D. The rules of indirect addressing
E. How to force branching with jumps and loops

V. How Subroutines are Placed in Procedures
A. How stack operations work
1. Using the stack to preserve register values
2. Using the stack to pass parameters
B. Procedures are defined using specific syntax
C. The call to a procedure
D. What to do when procedures are in an external library
E. What is a stack frame and how is it used
F. Performing recursion performing recursion performing recursion
G. INVOKE, the powerful procedure tool available in 32-bit x86 assembler

VI. How to Perform Conditional Processing
A. Comparison instructions create Boolean results
B. How to perform conditional jumps and loops
C. Using conditional processing to mimic conventional IF and WHILE branching
D. .REPEAT and .WILE directives

VII. Performing Arithmetic Operations
A. How and why to use shift and rotate
B. Multiplication and over ow
C. Division and remainder
D. Expanding simple addition with the carry instruction
E. Expanding simple subtraction with the borrow instruction
F. Arithmetic operations on decimal variables

VIII. Working with strings and Arrays
A. Defining strings and arrays
B. Using primitive string instructions
C. Building procedures that use strings
D. Operations on 2-dimensional arrays
E. How to build searching and sorting routines

IX. Using advanced constructs
A. How structures differ in assembler
B. Placing code in macros
C. Shortcuts using conditional-assembly directives
1. .IF .ELSE
2. .WHILE
3. .FOR

X. Integrating Assembler into Windows Programming
A. Creating Win32 Console programs
B. Creating graphical windows applications
C. Memory management and dynamic memory allocation
D. Inserting assembler with inline code
E. Building an object file in assembler

XI. Generating Assembler for the Co-processor
A. Understanding floating-point architecture
1. Binary representation
2. Floating-point stack
3. Unique registers and flags
B. Familiarization with the floating-point instruction set
1. Mixed mode assembler code
2. Arithmetic operations
3. Precision data types
4. Trigonometric functions