Study Guide for Exam 1

Chapter 1 – Basic Concepts

- *Section 1.1 (Welcome to Assembly Language) & Lecture 1* – Relationship between machine language and assembly language. Relationship between machine language and high-level languages. Assemblers and disassemblers. Compilers. What types of software are or are not written in assembly language, and why.

- *Section 1.3 (Data Representation), Lectures 2 & 3* – Binary representation; radix/base, bits, LSB, MSB, binary addition. Storage sizes. Hexadecimal representation. Two's complement representation. Conversion of integers between representations. Maximum and minimum representable values. ASCII and Unicode representation of character strings; UTF-8 encoding. Line endings (CRLF). Null-terminated strings.

Chapter 2 – x86 Processor Architecture

- *Section 2.1 (General Concepts in Microcomputer Design), Lectures 5–6* – CPU: registers, clock, control unit, ALU, memory storage unit. Data, I/O, control, and address buses. Clock cycles. Instruction execution cycle: instruction pointer/program counter; fetch-decode-execute (FDX) cycle, including fetching operands and storing output operands. Procedure for reading from memory; cache memory. Processes; tasks; multitasking; scheduler; task switching; preemptive vs. cooperative multitasking.

- *Supplemental: Executables, DLLs, and Loading – Lecture 6* – Executables (.exe files) vs. DLLs vs. static-link libraries; sharing of DLLs among many processes. PE file format: text, data, idata sections. Loaders; how EXE files are loaded; relationship between the text and data sections of a PE file and the code and data segments of a running process.

- *Section 2.2 (x86 Architecture Details), Lectures 6–7* – x86 modes of operation: protected vs. real-address modes. Basic program execution registers (EAX, AX, AH, AL, EBX, BX, BH, BL, ECX, CX, CH, CL, EDX, DX, DH, DL, EBP, BP, ESP, SP, ESI, SI, EDI, DI, CS, SS, DS, ES, FS, GS, EFLAGS, and EIP): general purpose, segment, flags, and instruction pointer registers; 32- vs. 16- vs. 8-bit register names; control vs. status flags. Accessing parts of registers. Why segment registers, EFLAGS, and EIP cannot be used for general-purpose computation.

Chapter 3 – Assembly Language Fundamentals

- *Section 3.1 (Basic Elements of Assembly Language), Lecture 7* – Integer constants, radix suffixes; integer expressions and operator precedence; character constants; string constants; reserved words; identifiers; directives vs. instructions; instruction format: label, mnemonic, operands, comments.

- *Supplemental: Basic Control Flow Using JECXZ – Lecture 8* – Unconditional vs. conditional jumps; JMP vs. JECXZ; translating do-while loops, while loops, and if statements.

- *Section 3.3 (Assembling, Linking, and Running Programs) –* Assemble-link-execute cycle. Linkers. Object files. Difference between static-link libraries and dynamically-linked libraries (DLLs).

- *Section 3.4 (Defining Data), Lecture 10 –* Intrinsic data types (Table 3-2). Data definition statements. Defining BYTE and SBYTE data; question mark (?) initializer; multiple initializers; defining strings; DUP operator. Defining WORD, SWORD, DWORD, SDWORD, QWORD data. Memory layout for arrays. Little vs. big endian. Declaring uninitialized data; .DATA? directive.

- *Section 3.5 (Symbolic Constants), Lecture 11 –* Equal-sign directive; current location counter ($). Calculating sizes of arrays and strings. EQU directive. TEXTEQU directive.

CHAPTER 4 – DATA TRANSFERS, ADDRESSING, AND ARITHMETIC

- *Sections 4.1–4.2 (Data Transfer Instructions, Addition and Subtraction), Lectures 12–14 –* Immediate, register, memory operands; notation. Direct memory operands. MOV instruction (data transfer); overlapping values (e.g., EAX, AX, AH/AL). Zero/sign extension of integers: overlapping registers vs. MOVZX vs. MOVSX, allowable operands for MOVZX and MOVSX. INC, DEC, ADD, SUB instructions. Carry, overflow, zero, sign, parity, and auxiliary carry flags; effects of arithmetic instructions on flags.

You should be prepared to write and analyze small assembly language programs using the instructions studied so far (MOV, MOVZX, MOVSX, ADD, SUB, INC, DEC, JMP, JECXZ), along with calls to ReadString, WriteString, ReadDec, WriteDec, ReadInt, and WriteInt. You will be asked to fill in the .code and .data sections; you will not need to write entire .asm files.

Many exam questions will be similar to questions from the in-class activities and homework assignments.

An ASCII table will be provided if it is necessary for the exam. Calculators will **not** be allowed (and should not be necessary).