(CSCI 365) Exam Two Review

Note: Exam Two covers Chapter 4, 5, and 6. The test will consist of two parts. Part I consists of MC/MA/TF questions, which come from the same pools of your homework. So review homework, and understand why the answer is correct, is very important. Part II consists of short answer questions. It code be coding, calculation (show work), or concepts. Make sure you understand how to do the sample questions at the end of this review. The real test questions may not be the same as the sample questions. However, know how to do sample questions will improve your real test grade for sure.

Basic Concepts

From Chapter 4 PPT Summary:

1. Data Transfer
    a. MOV – data transfer from source to destination
    b. MOVSX, MOVZX, XCHG
2. Operand types
    a. direct, direct-offset, indirect, indexed
3. Arithmetic
    a. INC, DEC, ADD, SUB, NEG
    b. Sign, Carry, Zero, Overflow flags
4. Operators
    a. OFFSET, PTR, TYPE, LENGTHOF, SIZEOF, TYPEDEF
5. JMP and LOOP – branching instructions

From Chapter 5 PPT Summary:

1. Procedure – named block of executable code
2. Runtime stack – LIFO structure
    a. holds return addresses, parameters, local variables
    b. PUSH – add value to stack
    c. POP – remove value from stack
3. Use the Irvine32 library for all standard I/O and data conversion
    a. Basic input/out functions, such as Write(/Read)Int(/Hex/Dec/Char/String), Crlf, Clrscr, RandomRange, SetTextColor, WaitMsg

From Chapter 6 PPT Summary:

1. Bitwise instructions (AND, OR, XOR, NOT, TEST)
    a. manipulate individual bits in operands
2. CMP – compares operands using implied subtraction
    a. sets condition flags
3. Conditional Jumps & Loops
    a. equality: JE, JNE
    b. flag values: JC, JZ, JNC, JP, ...
    c. signed: JG, JL, JNG, ...
    d. unsigned: JA, JB, JNA, ...
    e. LOOPZ, LOOPNZ, LOOPE, LOOPNE

4. Flowcharts – logic diagramming tool
5. Finite-state machine – tracks state changes at runtime

Sample Short Answer Test Questions

Note: 8-10 similar questions will show up in real test. All sample questions come from textbook Algorithm Workbench sections

1. Write a sequence of MOV instruction that will exchange the upper and lower words in a double-word variable named **three**
2. Using the XCHG instruction no more than three times, reorder the values in four 8-bit registers from the order A, B, C, D to B, C, D, A
3. Write a sequence of two instructions that use addition to set the Zero and Carry flags at the same time
4. Write a sequence of two instructions that set both the Carry and Overflow flags at the same time
5. Write a sequence of instruction showing how the Zero flag could be used to indicate unsinged overflow after executing INC and DEC instructions
6. Write a loop that iterates through a double-word array and calculates the sum of its elements using a scale factor with indexed addressing

   Use the following data definition for Question 7-9
   .data
   myBytes BYTE 10h, 20h, 30h, 40h
   myWords WORD 3 DUP(?), 200h

7. Write an instruction that moves the second byte in myWords to the AL register
8. Write an instruction that moves all four bytes in myByte to the EAX register
9. Insert a LABEL directive in the given data that permits myBytes to be moved directly to a 16-bit register
10. Write a sequence of statements that use only PUSH and POP instructions to exchange the values in the EAX and EBX registers
11. Write a sequence of statements using indexed addressing that copies an element in a double-word array to the previous position in the same array
12. Write a sequence of statements that display a subroutine's return address. Be sure that whatever modifications you make to the stack do not prevent the subroutine from returning to its caller
13. Write a single instruction that converts an ASCII digit in AL to its corresponding binary value. If AL already contains a binary value (00h to 09h), leave it unchanged
14. Write instructions that jump to label L1 when the unsigned integer in DX is less than or equal to the integer in CX
15. Write instructions that jump to label L1 when the signed integer in DX is less than or equal to the integer in CX

16. Implement the following pseudocode in assembly language. Use short-circuit evaluation and assume that val1 and X are 32-bit variables
    if(val1 > ecx AND ecx > edx) X = 1 else X = 2
17. Implement the following pseudocode in assembly language. Use short-circuit evaluation and assume that val1 and X are 32-bit variables
    if(val1 > ecx OR ecx > edx) X = 1 else X = 2