Name: ________________________________ **Lab Section:** _______

Objectives: To review important concepts in Chapters 3 and 4. Answer on this sheet where space is given.

References: ECOA2e Section 3.2.1-3.2.4, 4.1-4.6, 4.8.1-4.9.1, 4.9.3, 4.10, 4.11.1 and associated Chapter Slides
Class Notes (via course home page): **bit_operations.txt**, **text_errata.txt**

Not all questions will be marked – check all your answers against the answer sheet when it is posted.

1. What happens to the value of a binary number if you "shift" the bits to the left two places by adding two zeros after the rightmost binary digit, e.g. 11001_2 --> 1100100_2 ________________________________
2. What happens to the value of an octal number if you "shift" the number to the left one place by adding one zero after the rightmost octal digit, e.g. 0377_8 --> 03770_8 __________________________
3. What happens to the value of a hexadecimal number if you "shift" the number to the left one place by adding one zero after the rightmost hex digit, e.g. 0xABC --> 0xABC0 __________________________
4. True / False – decimal 1234.0×10^{37} fits in IEEE 754 single-precision floating-point.
5. True / False – decimal 0.00001×10^{40} fits in IEEE 754 single-precision floating-point.
6. Circle the values that fit in a 32-bit two's complement integer with no loss of range or precision:
   $2^{30}-3$ $2^{30}-1$ 2^{30} $2^{30}+1$ $2^{30}+3$ $2^{30}+2^{29}$
7. Circle the values that fit in IEEE 754 single-precision floating-point with no loss of range or precision:
   $2^{30}-3$ $2^{30}-1$ 2^{30} $2^{30}+1$ $2^{30}+3$ $2^{30}+2^{29}$ (*Hint: look at the binary mantissa*)
8. Express in hexadecimal the value stored in memory by each of the following C bitwise expressions:

   char x = ~0x1; ____________ char x = ~0x10; ___________ char x = ~0 & 0xAA; ____________
   int x = ~0x1; ________________________________ int x = ~0x10; __________________________
   int x = ~0 & 0xAA; ____________________________ char x = 0x11 | 0xAA; __________________
9. Give (hex) a bit mask that will mask off (zero) everything except a MARIE opcode: ________________
10. Give (hex) a bit mask that will mask off (zero) everything except a MARIE address: ________________
11. Give a C language expression that will turn an ASCII Control character into the corresponding ASCII lower-case letter: ____
12. How many address bits do you need to address byte-addressable 2Mx32 memory? _______________
13. How many address bits do you need to address word-addressable 2Mx32 memory? _______________
14. How many address bits do you need to address byte-addressable 4Mx16 memory? _______________
15. How many address bits do you need to address word-addressable 4Mx16 memory? _______________
16. Question 8, p.238: a) __________ b) ___________ c) _______________ d) ___________________

17. Question 9, p.238: a) ____________________ b) ______________________ c) ____________________
18. Memorize the names and functions of the seven MARIE registers on p.191. Write the full names of the registers here: ____

    ____
19. True / False – unlike MARIE, modern computers have multiple general-purpose registers. (p.192)
20. True / False – unlike MARIE, the ISAs of modern computers have hundreds of instructions. (p.193)
21. Memorize the meanings of the nine basic MARIE instructions in Table 4.2. Reproduce that table here:

    ____
    ____
    ____
    ____
    ____
    ____
    ____
    ____
    ____
22. Define an instruction "mnemonic" (p.195): ____
23. Another name for "binary instructions" is (p.195): ____
24. The mnemonics that correspond to machine code are referred to as: ____________________________
25. True / False – every assembly language instruction corresponds to exactly one machine instruction.
26. The name of the program that converts mnemonic assembly language to its binary equivalent machine code is (p.195, also Section 4.11): ____
27. Give the hex for "Skip if AC less than zero": ____
28. Give the RTL for "Add X": ____
29. Give the RTL for "Jump X": ____
30. Reproduce here the one-line descriptions (no RTN) of the *revised* instruction processing trio of operations from *revised* Section 4.9.1 (see the revised version in the Class Notes, file **text_errata.txt**):
    1. ____
    2. ____
    3. ____
31. Suppose the program in Table 4.3 started at hex address zero. Give the seven 16-bit hex values for the contents of memory: ____
32. An assembler reads a source file and produces as output (p.206): _________________________________