SOME REVIEW NOTES AND PRACTICE QUESTIONS
FOR FIRST EXAMINATION

• Number Bases and Conversions
You need to become familiar with numbers expressed in binary and hexadecimal. Be able to com-
fortably convert numbers from decimal notation to binary, binary to decimal, hexadecimal to binary,
binary to hexadecimal, decimal to hexadecimal and hexadecimal to decimal. You should get to the
point of being able to do these conversions quickly.

Problems
1. Convert the following decimal numbers to binary: 28; 13; 97; 143
2. Convert the following binary numbers to decimal: 1010; 1101; 111001; 1001100; 1111110
3. Convert the following binary numbers to hexadecimal: 101111100; 1101; 1011001110011;
   110100000100100
4. Convert the following hexadecimal numbers to binary: F57; C8; FFF3

• Signed Integers
You need to become familiar with encoding a range of integers in a fixed number of bits. Know
different representations: Signed Magnitude, Two’s complement, and Biased. Know how to represent
integers in all representations and know how to recognize the decimal value of numbers represented.
Know how to do arithmetic—addition and subtraction—in two’s complement, including knowing the
overflow rules. Know the range of numbers represented for a given size (number of bits).

Problems
5. Suppose a 7-bit representation. What are the decimal values of the following two’s complement
   binary numbers? 0001110; 1111000; 1110101
6. Suppose we are using 6 bits to represent two’s complement integers. Do the following additions.
   Also convert the addends and the sum to decimal to check your work. Indicate any overflow in
   the sums. 010101 + 111001; 001010 + 001011; 101111 + 101110
7. Using 8 bits to represent negative integers, show -71 in binary in the following representations:
   signed magnitude; two’s complement.
8. What is the range of numbers that can be represented with 6 bits using two’s complement repre-
    sentation?
9. Do the following subtraction using the method of adding the complement of the number to be
    subtracted: 011011 - 001101. Assume a 6-bit two’s complement representation.

• Floating Point Numbers
You need to know how to convert between decimal real numbers and binary, and how to represent
them in a normalized format of a given standard. Note that you are not required to know the details
of the IEEE 754 floating point standard but you are expected to be able to work with a given smaller
hypothetical standard, similar to the example we have done in the lecture.

Given a floating point representation, what is the smallest and largest positive representable number?
What is the smallest and largest gap between two successive representable numbers? What is the total
number of distinct numbers that can be represented?

Problems

10. Complete the following table using base 2 scientific notation and an eight-bit floating point rep-
resentation in which there is a three-bit exponent in biased 3 notation and a four-bit normalized
fraction with a hidden 1. In this representation, the hidden 1 is to the left of the binary point
and is not explicitly stored. This means that the number 1.0101 is in normalized form, whereas
.10101 is not.

<table>
<thead>
<tr>
<th>Base 2 scientific notation</th>
<th>Floating point representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sign</td>
</tr>
<tr>
<td>+1.1001 × 2⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>-1.1 × 2⁴</td>
<td>1</td>
</tr>
</tbody>
</table>

• Character Codes

Read about the various character codes. We will be using the ASCII code in programming.

• Logic Design, and Arithmetic and Logic Operations

Know that a particular logic function can be represented in two different ways: as a truth table, or as
a logic equation — they are equivalent. Know how to represent a logic function in both ways, i.e.,
given one way of representing a function, be able to give the other.

Understand the properties of Boolean Algebra. Know how to prove individual laws using truth tables.
Know how to use the properties to simplify Boolean equations.

Given a truth table, give the sum-of-products form.

You should be able to understand and write pseudocodes that use C/C++ bit-level and logical operators
to perform various functions.

• Assembly Language

Know the assembly programmer’s view and how it is different from the view of a C programmer.

Know the phases of program translation, the role of a compiler, assembler and linker.

Know the syntax and understand the semantics of Intel x86-64 instructions that we covered. You
should be able to trace the execution of a given (simple) assembly program, understand what it does,
and write an equivalent C code.

Problems

11. Write an equivalent C function for the following assembler routine. Note that the compiler
associates the (long int) argument of a (single-argument) function with the (8-byte) register
%rdi. And it associates the return (long int) value with (8-byte) register %rax.
foo:
    movq %rdi,%rax
    addq $4,%rax
    addq %rdi,%rax
    ret