Eric Roberts
CS 54N

Handout #8
October 13, 2016

Assignment #2

Due: Wednesday, October 19

Write out answers to each of these problems. You can write up your answers either on paper or using a word processor.

Problem 1—Binary representation

Answer all six exercises in the puzzle box on page 75.

Problem 2—Binary representation of perfect numbers

Greek mathematicians took a special interest in numbers that are equal to the sum of their ***proper divisors,*** which are simply those divisors less than the number itself. They called such numbers ***perfect numbers.*** For example, 6 is a perfect number because it is the sum of 1, 2, and 3, which are the integers less than 6 that divide evenly into 6. The next three perfect numbers—all of which were known to the Greeks—are 28, 496, and 8128.

Write out the binary representation of the first four perfect numbers. How would you describe the binary form of these perfect numbers? Although the proof is well beyond the scope of this class, it turns out that *all* even perfect numbers have this form; the question of whether any odd perfect numbers exist remains open in mathematics.

Problem 3—Binary arithmetic

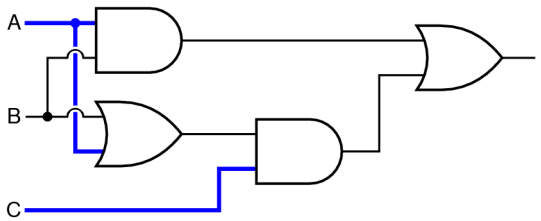
Complete the three calculations in the puzzle box on page 76.

Problem 4—Hexadecimal notation

Answer parts (a) and (b) of the puzzle box on page 80. We'll do part (c) in class.

Problem 5—Logic gates

Although the puzzle box on page 105 asks you for a complete analysis, tracing all possible signals through the majority circuit is tedious. Using thicker lines to indicate signals that have the value **1**, trace the signals that flow through the majority circuit only for the following arrangement, when **A** and **C** are on and **B** is off:



Problem 6—The Toddler machine

Answer all three parts of the exercises in the puzzle box on page 130.

Problem 7—Assembly language

Translate the following assembly language program into Toddler machine instructions, showing the contents of all memory addresses loaded by the program:

```
start:   INPUT   n
         LOAD    #0
         STORE   total
         LOAD    #1
loop:    STORE   i
         LOAD    n
         SUB     i
         JUMPN   done
         LOAD    total
         ADD     i
         ADD     i
         SUB     #1
         STORE   total
         LOAD    i
         ADD     #1
         JUMP    loop
done:    OUTPUT  total
         HALT

i:       0
n:       0
total:   0
```

You can, of course, simply type this program into the simulator and copy down the machine instructions, but you will understand things better if you do the translation by hand and then use the simulator to check your answer.

What output does this program produce if you run it and enter 5 as the input value in response to the first instruction? What value does this program compute in general?