EECS 370
Homework #2

Problem #1:

What are the maximum number of ROM storage bits required to implement a finite state machine with 3 inputs bits, 17 internal states, and 4 bits of output?

2^(3 + ceil(log2(17)))
2^(3 + 5)
2^8
256 entries in ROM

ceil(log2(17)) + 4
5 + 4
9 bits per entry

total bits => 256 * 9 => 2304 bits

Problem #2:

How many NAND gates are needed to design an 8-bit 3-to-1 MUX? You may only use NAND gates to build the MUX, but the NAND gates may have any number of inputs. (Show your work for partial credit.)

The answer to problem 2 is 34 gates.
4 gates each for 1 bit 3-to1-mux. for 8 bits, there are 4*8 = 32 gates.
However, there are also two NAND gates used to invert the 2 selection bits. (We need the complement of both selection bits, but we only need to form these once.)
So total number of NAND gates = 32+2 = 34

Problem #3

a) Calculate the worst-case delay of an 8-bit ripple carry adder made up of 8 full adders such that the delay of a full adder is as follows: Any change to the input to the circuit takes 200 ps for it to be reflected in the sum and 125 ps for it to be reflected in the carry out bit. The worst case delay is defined as the longest possible amount of time it could take for the correct output to appear after any adder input change.

b) Next, provide a general expression for the worst case delay of a N-bit ripple carry adder given the delays specified in part (a) of the problem.

3) a) 1075 ps.

b) 125(n-1) + 200

Problem #4:

**A) The Big Picture of page 299 mentions that bits have no inherent meaning.
Given the bit pattern:
10001111111011111100000000000000
what does it represent, assuming that it is**

a. a two's complement integer?

-1,880,113,152

b. an unsigned integer?

2,414,854,144

c. a single precision floating-point number?

959 * (2 ** -105)
-2.364117525e-29

d. a MIPS instruction?

```
100011 11111 01111 11000 00000 000000
  op    rs    rt    rd   shamt funct
```

Either of these are acceptable answers:
lw $15, -16384($31)
lw $t7, -16384($ra)

**B) This exercise is similar to part a), but this time use the bit pattern
00000000000000000000000000000000**

a. a two's complement integer?
0

b. an unsigned integer?
0

c. a single precision floating-point number?
0.0

d. a MIPS instruction?

```
000000 00000 00000 00000 00000 000000
  op    rs    rt    rd   shamt funct
```

Either of these are acceptable answers:
sll $0, $0, 0
sll $zero, $zero, 0

Problem #5

Given: two signed (2's compliment) integers, in memory locations identified by labels num1 and num2.

Task: Write an LC2K4 assembly program to load these numbers into registers, subtract num2 from num1, and store this result in register 7. (e.g. $7 = $1 - $2).

Hint: Recall our discussion on adders, and why we include a carry-in. You must define num1 and num2, e.g. your program should include:

num1 .fill <insert your favorite number here>
num2 .fill <another number you like>

```
        lw      0       1       num1
        lw      0       2       num2
        lw      0       3       one
        nand    2       2       2
        add     2       3       2
        add     2       1       4       result in 4
        halt
num1    .fill   74
num2    .fill   36
one     .fill   1
```