**CS 220      Sample Exam 2**

1. Show how to implement an ***and-gate*** using ***nor-gates*** *only*.

2. Multiply the two five bit unsigned binary numbers 10101(multiplicand) and 11001 (multiplier) using the multiply algorithm tracing the values of the multiplier, multiplicand, and product at each step.

3. Using the integer divide algorithm from class divide the eight bit unsigned integer **11101101** by **1100.** Trace the values of the quotient and remainder for each step of the algorithm.

4. Consider an adder that adds three bits **a**, **b**, and **c** and computes two outputs, a **carry** and a **sum**.
   a. Draw the truth table for the circuit.
   b. Show the full sum of products equation for the **carry** output
   c. Derive a reduced **carry** equation that minimizes the number of gates required to implement it.

5. How many selector wires are required for a *mux* that has **16** inputs?

6. What MIPS assembly instruction does the machine code **0x8fa40000** represent?

7. What is the machine code for the MIPS assembly instruction **addi $t1, $sp, -4**

8. Write a MIPS function **max_array** that returns the largest element of an array given that array address and the number of elements. Your function should obey the all procedure calling conventions.

9. Write a MIPS main program that creates a sample ten-element array using the values 7, 0, 2, 9, 4, 99, 23, 11, 15, 88 and calls **max_array** developed in the question above. You should obey all procedure calling conventions.

10. Consider the C program below that has a recursive function reverse that prints an array in reverse order. Implement this function as a recursive MIPS assembly function. Implement the main program as well in MIPS assembly.

```
#include <stdio.h>

// curr is the current position in the array
void reverse(int vec[], int curr, int length) {
  if (curr == length) return;
  reverse(vec, curr + 1, length);
  printf("%d\n", vec[curr]);
}

int main() {
   int vec[] = {1, 2, 3, 4, 5}; // put this in a .data section
   reverse(vec, 0, 5);
}
```

11. Consider the one-bit MIPS ALU below.
    a) How should we set all of the inputs on the ALU above to compute the nor of a and b on the Result

    b) To compute whether **a** is less than **b** what should we set all of the inputs to? If this one-bit ALU was ALU zero in a 32 bit ALU Where would the **Less** input come from?