UNCA CSCI 320
Final Exam Spring 2016
28 April, 2016

This is a closed book and closed notes exam. It is to be turned in by 5:30 PM.

Communication with anyone other than the instructor is not allowed during the exam. Furthermore, calculators, cell phones, and any other electronic or communication devices may not be used during this exam. Anyone needing a break during the exam must leave their exam with the instructor. Cell phones or computers may not be used during breaks.

If you want partial credit for imperfect answers, explain the reason for your answer!

Name:_____

Problem 1 (8 points) Two's complement addition (Exer 1.56)
Convert the following decimal numbers into 6-bit two's complement binary numbers and add them. Indicate whether or not the sum overflows the 6-bit result.

$27_{10} + 31_{10}$	$-16_{10} + -9_{10}$

Problem 2 (2 points) Twos complement addition (Homework 5)
Suppose `A`, `B`, and `S` are declared as follows in SystemVerilog.

```
logic [15:0] A ;
logic [15:0] B ;
logic [15:0] S ;
```

Suppose `A`, `B`, and `S` are being used to represent 16-bit twos-complement numbers and the following statement is executed

```
S = A + B
```

Write an expression that tests if that addition results in an overflow. (Think about how you made this test in Homework 5.)

Problem 3 (3 points) x86-64

Two rather interesting x86-64 instructions, which appeared in Homework 11, are given in the table below. Write a couple of sentences of about what each of these instructions does and why they are frequently used in compiler-generated code.

`addq $1, %rdx`
`movl %ecx, 4(%rdi,%rdx,4)`

Problem 4 (2 points) Range of numbers

What is the range of numbers that can be represented by seven-bit unsigned numbers?

What is the range of number that be represented by seven-bit signed (twos-complement) numbers?

Problem 5 (2 points) Contrasts and definitions

What is *the* significant difference between combinational and sequential circuits?

Problem 6 (3 points) The flip-flop

Name at least three common connections ("ports" in SystemVerilog) of a flip-flop and, with a brief phrase, explain how these connections are used.

Problem 7 (12 points) SystemVerilog assignment and operators

Assume the SystemVerilog variables `P`, `Q`, `R`, `S`, `T`, `U`, `V` and `W` are declared as follows

```
logic [2:0] P ;
logic [2:0] Q ;
logic [4:0] R ;
logic [4:0] S ;
logic [6:0] T ;
logic [6:0] U ;
logic [6:0] V ;
logic [8:0] W ;
```

What are their values after the following assignments are executed in an `initial` block? Use the blanks on the right to write your answers and show your work. Only write in as many bits as the variables needs. Leave the others blank.

If all else fails, use your knowledge of C and Java bitwise operators.

```
initial begin
  P = 2 ;
                        //  P = _ _ _ _ _ _ _ _ _
  Q = 3'b100 | 3'd5 ;
                        //      _ _ _ _ _ _ _ _ _
                        //      _ _ _ _ _ _ _ _ _
                        //  Q = _ _ _ _ _ _ _ _ _
  R = {Q, P} ;
                        //      _ _ _ _ _ _ _ _ _
                        //  R = _ _ _ _ _ _ _ _ _
  S = R ^ 6 ;
                        //      _ _ _ _ _ _ _ _ _
                        //  S = _ _ _ _ _ _ _ _ _
  {T[2:0], T[6:3]} = {R, S} ;
                        //      _ _ _ _ _ _ _ _ _
                        //  T = _ _ _ _ _ _ _ _ _
  U = (R + S) << 1 ;
                        //      _ _ _ _ _ _ _ _ _
                        //      _ _ _ _ _ _ _ _ _
                        //  U = _ _ _ _ _ _ _ _ _
  V = ~U ;
                        //      _ _ _ _ _ _ _ _ _
                        //  V = _ _ _ _ _ _ _ _ _
  W = V & 6'h2A ;
                        //      _ _ _ _ _ _ _ _ _
                        //  W = _ _ _ _ _ _ _ _ _
end
```

A FSM for the next few problems

Use the following SystemVerilog program for the next few problems.

```
module FinalFSM(input  logic clk,
                input  logic reset,
                input  logic a,
                output logic q);
  typedef enum logic {Y, Z} statetype;
  statetype state, nextstate;

  // state register
  always_ff @(posedge clk, posedge reset)
    if (reset) state <= Y;
    else       state <= nextstate;

  // next state logic
  always_comb
    case (state)
      Y:        nextstate <= Z;
      Z: if (a) nextstate <= Y;
         else   nextstate <= Z;
      default:  nextstate <= Z;
    endcase

  // output logic
  assign q = (state == Y);
endmodule
```

Problem 8 (5 points) FSM diagram

Draw an FSM diagram for **`FinalFSM`** in the space below:

Problem 9 (2 points) FSM tables

The SystemVerilog compiler will encode the state with one bit, which we will call `S`. When `S` is 0, the FSM is in state `Y`; and, when `S` is 1, the FSM is in state `Z`.

Give the encoded (or binary) state transition and output tables for `FinalFSM` below.

present state	input	next state
S	a	S'
0	0	
0	1	
1	0	
1	1	

present state	output
S	q
0	
1	

Problem 10 (2 points) FSM equations

Write the boolean equations corresponding to the tables you wrote for the preceding problem. These should be pretty simple.

Problem 11 (6 points) FSM digital logic

And finally, draw an implementation of the FSM using digital logic. There should be a four connection points and no more than three gates, along with a flip-flop.

Problem 12 (12 points) Caching

Consider a cache implemented with the following parameters

- 32-bit address are used to address the cache/RAM hierarchy
- The capacity of the cache is 32k bytes. That's 32768 bytes.
- Each block (or line) of the cache is 32 bytes.
- A 4-way set associative implementation is used.

12A: How many blocks (or lines) does the cache have?

12B: How many sets (or rows) does the cache have?

12C: The 32-bit address is divided into three fields: tag, index, and offset. How many bits are each allocated in each of these fields?

12D: What are the tag, index, and offset fields of the address `0xABBA2468`?

1010 1011 1011 1010 0010 0100 0110 1000

12E: Make a small drawing of the cache in the space below.

Problem 13 (12 points)

Their are eight boxes below and each contains a term or concept used in this course. Within the box, give a brief definition of that term. Your best six of eight will be graded. That is, you can skip two of them.

Branch prediction
Fully associative cache
Page table
Pipelining (in microarchitecture)
Propagate signal (Pi:j in carry-lookahead)
TLB – Translation Lookaside Buffer
Valid bit (in page table and cache)
Watch dog timer

Problem 14 (2 points)

Suppose you have a computer, like the IA-32 with PAE, with 32-bit virtual addresses and 36-bit physical address and 4k pages. How is the following virtual address divided into page number and page offset fields?

1010 1011 1011 1010 0010 0100 0110 1000

Problem 15 (2 points) Effective access time

In a computer with cached memory, suppose it takes 2 clock cycles to read data from the cache and 200 clock cycles to read data from RAM. Further suppose the average hit rate is 95%. What is the effective access for reading memory? Assume that an attempt is made to read the cache before an access is made to RAM.

Problem 16 (3 points) Array computations

The following two loops only differ in their bold array accesses. However, one should be much faster than the other. Why is that is the case?

```
sum = 0 ;
for (int i=0; i<1024; ++i) {
  for (int j=0; j<1024; ++j) {
    sum = sum + V[i][j] ;
  }
}
```

```
sum = 0 ;
for (int i=0; i<1024; ++i) {
  for (int j=0; j<1024; ++j) {
    sum = sum + V[j][i] ;
  }
}
```

Problem 17 (12 points)

In olden days, every Ethernet packet was preceded by a *preamble* that consisted of 31 alternating pairs of 1 and 0 and then two consecutive 1's. The 64 bits looked like this:

10101010101010101010101010101010101010101010101010101010101010101011

Your task is to write, in SystemVerilog, a preamble recognizer. It should output a 1 only if the preceding 64 bits matches the preamble seen above. Keep in mind that there could be more than 31 pairs of 10 (and this often did happen) and that the following sequence with 66 10's followed by an 11 is also an acceptable preamble.

10101010101010101010101010101010101010101010101010101010101010101010101010101011

You could solve this problem with a FSM with 64 states, but that would just bore and depress every one. Try for a solution with a small number of states and a counter.

The outline of a SystemVerilog module to implement the preamble checker is given on the next page. Complete the module to solve this problem.

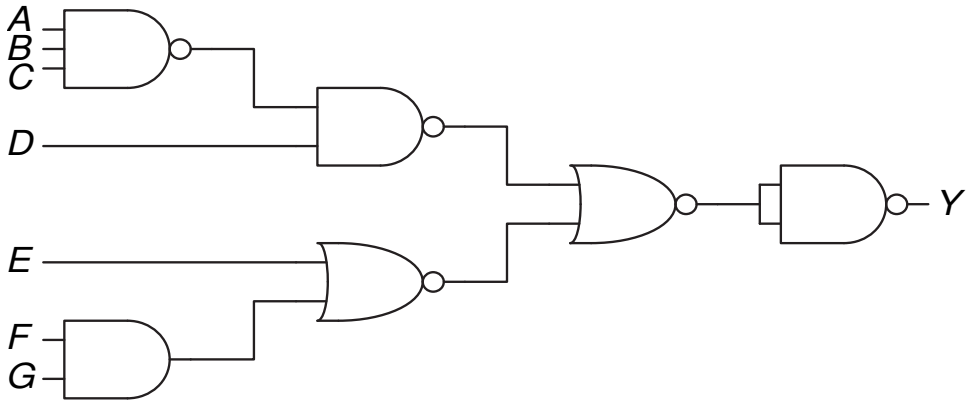
```
module ieee802preamble(input  logic clk,
                       input  logic reset,
                       input  logic a,
                       output logic q);
  // declare local variables here




  // do transition here
  always_ff @(posedge clk, posedge reset)
    if (reset)


    else




  // get ready for the transition here
  // use assigns or always_comb









  // output here
  assign q =


endmodule
```

Problem 18 (10 points) Contamination and propagation delays (Exercise 2.44)

The table below gives the contamination and propagation delays of several gates.

Gate	Propagation	Contamination
NOT (inverter)	15 ps	10 ps
Two-input NAND	20 ps	15 ps
Three- input NAND	30 ps	25 ps
Two-input NOR	30 ps	25 ps
Two-input AND	30 ps	25 ps
Two-input OR	40 ps	30 ps

What is the propagation and contamination delays of the two circuits shown below? Show your work and calculations for either full or partial credit



textbook figure 2.85