ECE199JL: Introduction to Computer Engineering **Fall 2012**
Homework 11 **Due: at start of lecture on Friday 9 November**

Von Neumann Computer Organization

Please do problems 4.4, 4.8, 4.10, and 4.16 from the textbook.

Here are 3 additional problems for this week:

1. Counter-Based FSMs
Our goal is to design a traffic-light controller with the following properties; it lights up the green light (output G) for 25 seconds, followed by the yellow light (output Y) for 5 seconds, then the red light (output R) for 30 seconds. The controller will repeat this pattern forever. Our system will have a clock signal with a period of 5 seconds.

A. How many states, N, are needed in a FSM to implement this controller? In how many of these states will the light be green, yellow, and red, respectively? What is the minimum number of state bits, n, needed to implement this controller?

B. Draw the state diagram for a FSM implementing this controller. Label states in which the light is green as GREEN1, GREEN2, etc., yellow as YELLOW1, etc., red as RED1, etc.

C. You will implement this FSM using a special down-counter that is available to you as a "black-box" circuit element. This n-bit counter (of the same minimum size you determined in Part A) has the following properties: it normally counts down in binary by 1 each clock cycle; (2) after counting down to 0 (and holding that for the usual 1 clock period) it parallel loads as the next state the bit pattern L[n-1],...,L2,L1,L0, and then continues counting down from there. Otherwise, it ignores the Lx inputs. This device thus has outputs Q[n-1],...,Q2,Q1,Q0, and inputs CLK and L[n-1],...,L2,L1,L0. In your solution, draw it as a rectangular box with CLK (positive-edge-triggered) entering from the top, the Lx inputs entering on the left side, and the Qx outputs on the right side.

Assign the highest-numbered needed binary values ($N-1$, $N-2$, ... to your green-light state(s), the next to your yellow state(s), and the lowest (...,2,1,0) to the red-light states. Based on this assignment, write the Karnaugh-maps for the three outputs G, Y, and R (make a fail-safe choice for any unused state-bit patterns, and briefly explain why you chose what you did), and sketch the circuit diagram for your solution, using the special down-counter as a circuit element that is already given to you; draw it as described above. (Points will be deducted for not using or not drawing this element as described!)

2. Memory Construction
Large memories are often constructed from a number of RAM chips of smaller size.

Print Figure 11.1 (posted separately on the course website), and sketch the combinational logic needed to construct a 64 by 4-bit memory ($2^6 \times 4$) memory from four separate (32×2) memory chips. The EM ("Enable Memory") signal on the microprocessor indicates that an external memory read/write is intended and that the address is ready; it will remain hgh through completion of the read/write operation. R/W' is high (1) to indicate a read operation, and low (0) indicates a write; a5-a0 and d3-d0 are the address and data bit lines, respectively. Address and data lines and R/W' are labeled similarly on the memory chips; the memory chips will ignore address and data bus activity and tri-state their outputs when their individual CE ("Chip Enable") signals are low, and will initiate a read or write sequence when their CE is high.

3. Memory-Mapped I/O Construction
Many computers use "memory-mapped I/O", in which a read or write to a small fraction of the total address space is mapped to the I/O devices rather than to the computer memory. Here you will design some of the combinational logic to implement such a system.

In Figure 11.2 (posted separately on the course website for easy printing), the memory range x38 to x3F is set aside for I/O; there is one input device at address x38, and a (different) output device at the same address (x38). Sketch all of the connections and combinational logic needed to make this system work correctly. The microprocessor and memory signals and d7-d0 on the I/O devices work as stated in Problem 11.2; the DE ("Device Enable") signals on the input and output devices will cause them to write or read data on the bus, respectively, when DE = 1; and tri-state or ignore the data bus, respectively, when DE = 0.

It is very important that at **most** one device drives the data bus at any given time, lest you burn out the data outputs of the chips! Make sure that your logic avoids activating multiple devices.