Jonathan W. Valvano First:_____ Last:_____
October 2, 2002, 12-12:50pm

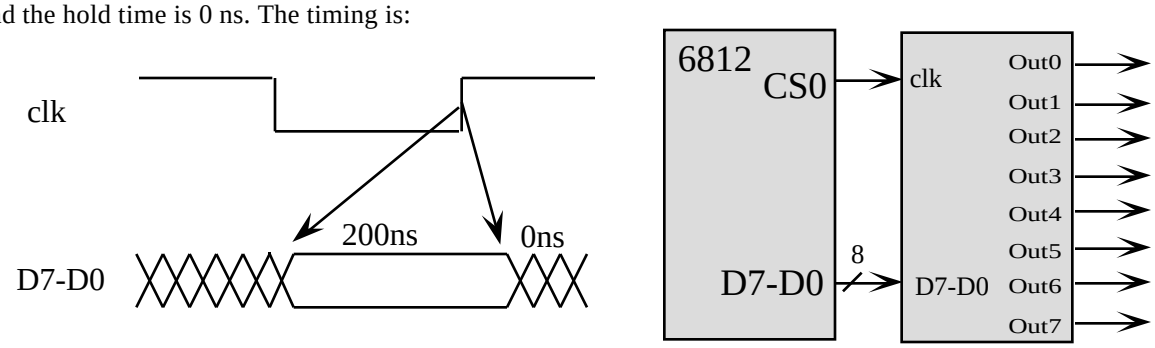
This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 50 minutes, so please allocate your time accordingly.
Please read the entire quiz before starting.

(35) Question 1. The purpose of this question is to perform a debugging **profile** on an existing software system. The existing software system consists of four major components: `Initialize()`, `Stuff()`, `Thingy()` and `Bailiwick()`. The static performance of this system is acceptable (i.e., the correct outputs are generated for each possible input value), but it runs too slow. After analyzing the data flow graph, you conclude the system is CPU bound. Your boss says rewrite the entire software system in assembly to make it run faster. You suggest she buy you a better compiler or a faster microcomputer, but she refuses. You compromise by suggesting a performance debugging technique to identify which of the four components uses the most execution time. Rewriting in assembly the component that uses the most time will have the biggest impact on software speed. To make a long story short, in the question you are to add minimally intrusive debugging instruments to determine which component requires the most execution time. PORTA and the SCI port are not used by the existing system. You may assume TCNT has been enabled to count at 1 MHz. You may also assume that each individual execution of one of the components takes less than 16 ms, but the total accumulated sum will be greater than 16 ms. You may develop any method you wish using the available hardware/software components in our lab. If you need hardware components like the SCI output, BDM, or 2-channel oscilloscope explain how they are to be used. You will add your software debugging instruments in between C statements of the existing program without modifying any of the existing software.

```
void main(void){
  Initialize();
    while(1){
    Thingy();
     if(Mode){
       Stuff();
    }
  }
}
#pragma interrupt_handler OC5handler
void OC5handler(void){
  Bailiwick();

}
```

(35) Question 2. Interface the following Output chip directly to the MC68HC812A4 data bus (not to an output port). The chip is write-only, and you may assume there are no read cycles to this address. There is one 8-bit output port on the chip. You will use the built-in CS0 built-in address decoder. For example, a write to 0x0200 will set 8 output lines. Assume an 8 MHz E clock. The rise of **clk** causes the 8-bit data to be latched into the chip. The setup time is 200ns and the hold time is 0 ns. The timing is:



Part a) What is the write data available interval? Express your answer as a function of the E clock period. Let t_{cyc} be the E clock period.

Part b) What is the write data required interval? Express your answer as an equation using only the terms like ↓**clk** and ↑**clk**. Don't calculate (yet) the actual interval in ns.

Part c) What is the smallest possible number of cycle stretches for this interface? SHOW YOUR WORK.

Part d) Draw the write-cycle timing diagram for the new interface. You may add any additional signals to clarify the system operation.

E

clk= CS0

Write Data Available

Write Data Required

(30) Question 3. Specify whether each statement is TRUE or FALSE.

	True/False
a) A **signed** `char` variable can store values from –128 to +128. E.g., `char myData;`	
b) Consider an **input** device. The **interface latency** is the time from when the software asks for new data until the time new data is ready.	
c) Consider an **output** device. The **interface latency** is the time from when the software sends new data to the output device until the time the output operation is complete.	
d) The `static` qualifier is used with functions to specify the function is permanent, created at compile time and is never destroyed. E.g., `static int AddTwo(int in){ return in+2;}`	
e) The `static` qualifier is used with a variable defined inside a function to specify the variable is permanent, created at compile time and is never destroyed. E.g., `void function(int in){ static int myData;`	
f) The `const` qualifier is used with a global variable to specify the variable should be allocated in ROM. E.g., `const int myData=5;`	
g) Code that is **friendly** means it can be executed by more than one thread with causing a crash or loss of data.	
h) Using the `SCI_OutString` and `SCI_OutUDec` for debugging is always **intrusive**.	
i) When interfacing memory to the 6812 in **expanded narrow mode**, PORTD contains the D7-D0 data bus.	
j) The `volatile` qualifier is used with variables to tell the compiler that code that accesses this variable should be optimized as much as possible.	
k) A read-modify-write access to a shared global variable always creates a **critical** section.	
l) The compiler automatically places an `sei` instruction at the beginning of the **interrupt service routine** and a `cli` at the end so that the computer runs with interrupts disabled while servicing the interrupt.	
m) In the **linked allocation** scheme described in the Lab 25 assignment, **internal fragmentation** occurs when entire large block is allocated to store a very small file.	
n) With a linked allocation scheme free blocks may be scattered in physically non-adjacent positions on the disk. Because the entire list of free blocks can always be linked together into one large file, the **linked allocation** scheme has no **external fragmentation**.	
o) Because the MC68HC812A4 has 22 address lines in extended mode, the DPAGE addressing scheme can access up to 4 million bytes of extended memory.	