Jonathan W. Valvano First Name: _______________ Last Name:____________________

May 17, 2010, 9am-12noon

Closed book part

(10) Question 1. Assume the Arm Cortex M3 is running with interrupts enabled using the PSP.

Part a) Which registers are pushed on the stack when an interrupt occurs?

Part b) Which stack pointer is used during the execution of the ISR?

Part c) What is in R14 (LR) while the computer executes the ISR? More importantly, what does this value mean?

Part d) Can one ISR interrupt another ISR? If so, explain under what conditions it can occur.

Part e) Does the STM32 employ hardware or software interrupt acknowledgement? Give one specific example of interrupt acknowledgement on the STM32.

(15) Question 2. List the four design choices one must make when implementing a spectrum analyzer (ADC->FFT->graphics display) like that generated in Lab 4. For each of the four choices, explain what the consequences are of that choice. I did not mean circuitry in front of the ADC, such as anti-aliasing analog low pass filter, analog circuit gain, analog input impedance,

1) Design choice and what does it mean?

2) Design choice and what does it mean?

3) Design choice and what does it mean?

4) Design choice and what does it mean?

(5) Question 3. Give the equation defining the Discrete Fourier Transform (DFT). Let x(n) be the sampled data, and make X(k) be the DFT output.

(5) Question 4. Consider a PID control system for a linear system.
(2) Part a) At what rate should you run the controller?

(2) Part b) In 16 words or less, explain the consequences of input noise.

(1) Part c) In 16 words or less, explain the consequences of time delay in the controller.

(5) Question 5. There are two threads that use bits in Port B as debugging monitors. A preemptive OS switches between threads. One thread sets Port B bit 11 by executing

```
   GPIOB->BSRR = 0x0800;   // make PB11 high
```

and the other thread sets bit 10 by executing

```
   GPIOB->BSRR = 0x0400;   // make PB10 high
```

The assembly instructions in both threads are similar to the following

```
  MOV  r0,#0x800     ;r0 = $0800
  LDR  r1,[pc,#932]  ;r1 points to GPIOB->BSRR
  STR  r0,[r1,#0x00]
```

Do these monitors constitute a critical section? Justify your answer.

(10) Question 6. In 16 words or less give a definition of the following terms

Part a) Internal fragmentation

Part b) Back EMF

Part c) Paging

Part d) Aging

Part e) Dual address DMA

Part f) Stuff bits

Part g) CAM

Part h) Input membership set

Part i) Hook

Part j) Bounded waiting

Jonathan W. Valvano First Name: _______________ Last Name:____________________

Open book part

Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). Please don't turn in any extra sheets.

(30) Question 7. You are given a CAN network, like the one developed in Lab 6. The CAN will be dedicated for the sole purpose of implementing one distributed spinlock counting semaphore. I.e., the CAN will not be used for other purposes except for implementing this one semaphore. You may assume the CAN is initialized on all computers, and the OS has access to the CAN via these two functions. The `CAN_ReceiveMessage` returns 1 if there are no messages waiting. If there is a message, `CAN_ReceiveMessage` returns the data and id by reference and returns 0. You may assume `CAN_SendMessage` always returns 0, meaning the message has been queued for sending.

```
int CAN_ReceiveMessage(unsigned short *data, unsigned short *id);
int CAN_SendMessage(unsigned short data, unsigned short id);
```

The above CAN programs send 16 bits of data. Feel free to increase or decrease the amount of data as you wish. You may assume only one thread per computer calls wait and only one thread per computer calls signal. You are allowed to make additional simplifying assumptions if you need to. You may use, without showing implementation, any OS function you implemented in your Lab 2 or Lab 3 OS. It is important not to introduce critical sections. Implement `OS_InitNetSem` on the computer containing the official copy of the counter. Explain how you use the CAN id/data.

```
long Counter;  // official copy of semaphore
// ******** OS_InitNetSem ************
// initialize network semaphore
// inputs:  initial Counter value
// output: none
void OS_InitNetSem(long value){
```

Implement these two functions also on the computer containing the official copy of the counter.

```
// ******** OS_LocalWait ************
// if Counter>0, decrement; if Counter is zero, spin
// input:  none
// output: none
void OS_LocalWait(void){
```

```
// ******** OS_LocalSignal ************
// increment Counter
// input:  none
// output: none
void OS_LocalSignal(void){
```

These two functions are called on a computer not containing the official counter.

```
// ******** OS_NetWait ************
// if Counter>0, decrement; if Counter is zero, spin
// input:  none
// output: none
void OS_NetWait(void){
```

```
// ******** OS_NetSignal ************
// increment Counter
// input:  none
// output: none
void OS_NetSignal(void){
```

(20) Question 8. Implement a memory manager for fixed sized blocks. Let the block size be

```
#define SIZE 100   // size in bytes
```

Let the number of blocks be

```
#define NUM 10     // number of blocks
```

For these two definitions, 1000 bytes of memory will be managed. Create three functions: initialization, allocate and deallocate.