Jonathan W. Valvano February 25, 2011, 11:00 to 11:50am

(10) Question 1. A real-time system executes two periodic tasks.
Part a) This the time jitter, Task 1 is delayed by Task 2.
Part b) Task 2 has priority, because it is not delayed.

(10) Question 2.
Part a) This implementation of the function is reentrant because it uses registers and has no writes to shared global variables. The compiler optimized away the `static`.
Part b) The input parameters `n` and `m` are passed in R0 and R1.
Part c) LR contains the return address (old PC) during the execution of the function.

(10) Question 3. These two `#define` definitions are bit-banded addresses for Port F.
Part a) These codes do not have critical sections because of the bit-banded addressing. This means the ISR only changes bit 0 of Port F and the other function only changes bit 1.
Part b) The compiler chose R0, R1 because they are automatically saved on the stack during an interrupt context switch, while R4 and R5 are not saved.
Part c) The bottom four bits of LR are 1001 during the execution of this ISR, LR=0xFFFFFFF9.

(10) Question 4. For each term, give a definition in 20 words or less.
Part a) Bounded waiting. The condition where once a thread blocks, there are a finite number of threads that will be allowed to proceed before this thread is allowed to proceed.
Part b) Atomic. An operation, once started will run to completion without interruption.
Part c) Slack time. Let t1 be the time until the deadline of a Task. Let t2 be the remaining time it will take to completion the Task. Slack time is t1-t2.
Part d) Hook. An OS feature that allows the user to run user-defined software at specific places within the OS. These programs are extra for the user’s convenience and not required by the OS itself.
Part e) Stabilization. A debugging technique that allows you to fix the input conditions (values and arrival times) so that a system can be run over and over generating the exact same outputs. This way, when you change the software, you know the change in outputs is the result of the change in software and not because of a change in input.

(10) Question 5. No deadlock can occur because the semaphores are numbered and each thread requests the threads in numerical sequence. This means there can be no circular wait in the resource allocation graph.

(25) Question 6. Implement an OS function that will wait on two semaphores.

```
void OS_Wait2(long *sema1Pt, long *sema2pt){
  OS_DisableInterrupts();     // Test and set is atomic
  while((*sema1Pt <= 0)||(*sema2Pt <= 0)){  // disabled
    OS_EnableInterrupts();
    OS_DisableInterrupts();
  }
  (*sema1Pt)--;             // disabled
  (*sema2Pt)--;             // disabled
  OS_EnableInterrupts();    // disabled
}                           // enabled
```

(20) Question 7. Consider an operating system that employs a priority scheduler.

Part a) Show the C code that implements priority scheduling.

```
nextPt = StartPt;
while(nextPt){  // list not empty
  if(nextPt->Sleep==0) break;
  nextPt = nextPt->Next; // find highest priority not sleeping
}
```

Part b) Show the C code that enters the TCB into the linked list in priority-sorted order.

```
if(StartPt){  // list not empty
 // special case: new will be first
 if(StartPt->Priority>newPt->Priority){
    newPt->Next = StartPt;  // chain as first
    StartPt = newPt;
 } else{
   pt = StartPt; // search for TCB before
   while(pt->Next){
    if((pt->Priority       < newPt->Priority)&&
       (pt->Next->Priority > newPt->Priority)){
      break; // success
    }
    pt = pt->Next;
   }
   newPt->Next = pt->Next; // chain in place
   pt->Next = newPt;       // pt points to TCB in front of new
 }
} else{
 StartPt = newPt;
 newPt->Next = 0; // first and only in chain
}
```