RULES OF THE EXAMINATION.

1. Do question 1, one of questions 2 and 3, and one of questions 4 and 5.
2. You must work independently. Phoning your partner to find out what is in your kernel is not considered independent.
3. You may use any source of information you want on this examination. Any information from sources you consult MUST be referenced. (Your memory, course notes and lectures are the only exceptions.)
4. I prefer answers in PDF format (whatever.pdf). If PDF is inconvenient then I accept plain text (whatever.txt) but you will have to stretch a little to make diagrams. Three pages (~1000 words, or less if there are diagrams) is as long an answer as you need for any question, but only if you write the appropriate three pages. Regurgitating the question or course notes get few or no marks. (See 7., below.) Put your name, student number and userid on every page.
5. Your answers should be submitted by e-mailing them to me at wmcowan@cgl.uwaterloo.ca.
6. A strategy that worked well for me as a student was to read the exam twice, then do something else for a couple of hours, then plan my answers, rest again, and finish by writing them.
7. Please remember that the questions are open-ended: you get most of your marks from going beyond simple answers; explicit instructions are intended as prompts to get you started in the right direction. Answering only what they request gets you about half the available marks for a question. The other marks come from ideas that go beyond the question as asked. You gain marks for the thoughts that you contribute to your answers. Write other people's thoughts, including mine, only to the extent that I need them to understand your answer.
8. When the examination says 'your kernel' it means the kernel you actually created, not an ideal kernel or the kernel you wish you had created. When the examination says 'your OS' it means your kernel plus the other tasks (couriers, notifiers, servers) on top of which applications run. When the examination says 'your train application/project' it means the application you planned to create, in what you would consider to be its final form.
9. All measurements and estimates must have units. If your unit is ticks translate it into milliseconds using the size of tick in your kernel.
10. There may be places in the examination where you must make assumptions. Do so, being certain that you explain your assumptions and how they are related to what you are saying.
11. Read each question carefully, and more than once. More marks are lost because of misunderstood questions than from any other single cause.
12. In all questions you should give your reasoning. More marks are given for reasoning than for correctness.
13. The cover page exists only to fulfil the registrar's regulations.
14. Read 7., above, once more.

* To understand why I handle the exam as described please consult the Introduction to the course.
DO QUESTION 1, ONE OF QUESTIONS 2 & 3, AND 

ONE OF QUESTIONS 4, & 5.

Question 1. Storing Register Values on Context Switches

When you call a sub-procedure, it may use registers that are in use by the calling procedure. Such registers must be stored before the sub-procedure executes and restored afterwards. Either (1) the calling procedure, the caller, may store them, or (2) the called sub-procedure, the callee, may store them, or (3) some may be stored by the caller, others by the callee.

1.a. Function calls. The compiler used in the course uses the mixed strategy (3), minimizing the total amount stored.
   (i) Which registers are stored by the caller and which by the callee?
   (ii) When optimizing code for speed we try to put as many local variables as possible into registers (Why?), and we try to save as little state as possible on procedure calls (Why?). Explain how these two objectives pull in different directions.
   (iii) Suppose you are hand-optimizing. How would you manage the above trade-off in practice? What would be the bad effect on the footprint of the executable?

1.b. System calls. In your kernel, when a task makes a kernel request (aka system call, request for service, etc.) register values must be stored and restored. In this part consider the task to be the caller, the kernel to be the callee and assume that there are no hardware interrupts.
   (i) Which storing strategy does your system use? Which registers are stored by whom, and why?
   (ii) Could the callee store only the registers it uses, restoring them on exit? Explain how it could be done, or why it can’t be done?
   (iii) Could the caller store only the registers it uses, restoring them on exit? Explain how it could be done, or why it can’t be done?

1.c. Hardware interrupts. In your kernel, when a hardware interrupt occurs there are several candidates for caller and callee: the kernel, the interrupted task, the task that called AwaitEvent, the hardware and nobody.
   (i) Select the identification that best suits your kernel and describe the register storage that occurs in your kernel.
   (ii) Read the description of the fast interrupt (FIQ) in the ARM documentation, and of the registers available in FIQ mode. Can you think of a way to use those registers as hardware caller or callee? If so describe your idea; if not describe why it can’t be done.

1.d. Dirty bits. Modern hardware contains many registers full of dirty bits. In a cache, for example, a dirty bit is set by hardware each time there is a write to a cache line. Then, when a line is flushed from the cache it is written back only if its dirty bit is set.

In this part of the question suppose you have an ARM processor exactly like the one used in the course, augmented by one 16-bit register, r16. It is accessible by privileged instructions: writing r16 resets all the bits, reading r16 gives you its current state. Every time a register in the CPU is written the corresponding bit of r16 is set. Thus, at any time in the execution it lets you know exactly which registers may have changed value since you last reset r16.
   (i) Describe how you might use r16 to economize on the amount of data you need to store on hardware and software context switches.
Question 2. The CLZ Instruction.

The ARM v5 instruction set contains a very useful instruction documented in the Programmer’s Guide but not implemented on our CPUs, which have only the v4 instruction set. The instruction is CLZ, count leading zeros, which counts the leading zeros in the value in the source register and loads it into the destination register.

2.a. Scheduling. In the lectures I mentioned CLZ in connection with scheduling.

(i) Explain how you could use CLZ to get insertion into and extraction from priority queues that is constant time with respect to the number of priorities. Describe the problem cases and how CLZ solves them.

(ii) Give pseudocode for your scheduler using CLZ, including the exact assembly language instructions that set and reset bits in the bitmask, in addition to the CLZ instructions themselves.

2.b. Other things. They wouldn’t include an instruction that was useful only for writing schedulers. (Or would they?)

(i) Think about the type of programming problems that would benefit from CLZ. Give the features they must have in common. (Hint. If the benefit of something is small it must be used many times to be worth having.)

(ii) Either, give examples of problems where CLZ is useful, or explain why there are no such problems. (Hint. Remember that there exist programs in problem domains that a greatly scaled up from your project, such as a Clock server with thousands of clients.)
Question 3. Memory Protection for your Kernel.

A multi-user operating system acts as though its users might be, and sometimes are, incompetent or hostile. Its current method for doing so is hardware protection. Tasks wishing to access memory require special permission, granted by the operating system, to access it. Trusting you partner(s) and yourself, you didn’t have to worry about incompetent or hostile users, and a good thing too. The normal method for enforcing operating system protection from other users takes advantage of special hardware, usually a memory management unit (MMU), which would take you at least an extra week to get operating adequately.

3.a Handling an Illegal Access. When the MMU detects a prohibited memory request it causes a data abort exception, which should trigger code that cleans up the problem. If your program tries to access protected memory Unix normally terminates it with the cryptic message ‘Segmentation violation’.

(i) Describe, in the amount of detail that seems appropriate to you, what needs to be done when a data abort exception occurs and how you would implement it.

3.b Care and Feeding of a Memory Management Unit. An MMU needs lots of tables, enough that DRAM is the only place to keep them. But DRAM is too slow.

(i) If you were to have provided separate protection for every task in your project, how big would be the tables for the ARM MMU that we used? What is in the tables? Why is DRAM too slow? Hint. Think about how the ARM MMU handles bigger and smaller chunks of memory.

To speed up access to the tables the MMU has a cache, the table lookaside buffer (TLB).

(ii) What happens to the TLB when a context switch occurs? Estimate the cost using the temporal unit of your choice. What could be done in your context switch to minimize the cost?

(iii) How many tasks could be protected if you want never to eject a page from the TLB?

(iv) When is the benefit of memory protection worth its cost? (Hint. When deciding to accept the high cost you ask if the cost of not having it is higher.)

* Based on prior experience in the course.
† See rules 3 and 6 in Rules of the Examination.
Question 4. CPU Congestion.

One rule of thumb in the trains project is that the idle task should be running more than 90% of the time. This question explores the reasons behind that recommendation.

In general, we like the CPU to be inactive when an interrupt occurs so that the action needed in response to the interrupt occurs as quickly as possible. It is, of course, impossible to ensure this condition because the sources of interrupts are not synchronized. Therefore, the CPU is occasionally busy processing a previous interrupt when a new interrupt occurs.

Thus, the execution profile of your project resembles something like the time-line to the right, with the CPU 100% busy for a short time after each interrupt occurs. (The figure shows two activities running in response to periodic interrupts. Assume that the periods are incommensurate and that the computation will conflict from time to time.) In each burst of activity the first part is the running of high priority tasks associated with low-level input processing, followed by lower priority tasks that integrate the input into the state of the program and calculate the response to the interrupt. Transmitting the response through output devices then occurs, handled by periodic high priority tasks. In this question we try to give this vague description a quantitative foundation.

As mentioned at the beginning of the exam, your answer should be based on a reasonably complete version of your project, such as a complete train control 2 result.

4.a. Enumerating interrupts. Most of the interrupt processing in your project occurs in response to one of six interrupts:

1. input ready from the train UART,
2. transmit buffer empty from the train UART,
3. modem flags changed from the train UART,
4. input ready from the terminal UART,
5. transmit buffer empty from the terminal UART, and
6. counted through zero from the timer.

For each of these interrupts, estimate

(i) the maximum frequency at which it can occur (in occurrences per second),
(ii) the average frequency at which it occurs (in occurrences per second),
(iii) an estimate of the amount of processing it requires (in microseconds), and

1. For each answer give reasoning and/or calculations to support your answer.
2. You may want to apportion your answer among several different computations called for by the interrupt. For example, when transmitting train commands, there may be a significant difference between the first byte of a command and the second, or between a sensor poll command and a train speed command. If you do so give the answer as “x% this, y% that”.
3. The numbers you give should be consistent with the idle time measurement for your project.

4.b. Coinciding interrupts.

(i) Using the train transmit UART interrupt and the terminal input UART interrupt estimate the increase in response time given the priorities described above. Explain your answer as quantitatively as possible.
(ii) The train transmitter UART might be transmitting a switch command to a turnout. How much competing CPU usage would be necessary to delay switching by 50 milliseconds?
Question 5. Making one Train Follow Another

In the past few terms several groups have made projects in which one train followed another very closely, without hitting it from behind and without getting too far behind. For example, several terms ago a group did a project in which trains could, at random times and places, be disabled on the track. Then a rescue train went out to bring it back. The rescue train approached the disabled train closely, then both went together to a siding, the disabled train leading, the rescue train following.

5.a Two trains travelling as one. When two trains travel as one, sensor attribution is tricky. Each sensor is hit twice in quick succession, once by the leading train, once by the following train.

(i) What is the time interval between the sensor hits? Give times in milliseconds for several speeds, using measurements from your project.

(ii) Suppose you detect the time interval by continual sensor polling, and time-stamp each byte as you receive it from the train controller. What intervals would you measure between successive time-stamps? Give the range of true time intervals you should associate with each measured value. (Numbers from your train project please.)

(iii) Suggest an alternative method of measuring that would estimate the intervals more precisely.

5.b Maintaining equal velocities. When the following train gets too close you must slow it down a bit, and when it gets too far behind you must speed it up a bit.

(i) How can you determine that the following train is too close or too far? Estimate the error in the measurement you make of distance.

(ii) Suppose the distance was the only thing that mattered. Then you could devote limitless computation to minimizing the error. How would you go about that? (You are allowed to modify software in any way you like but you are restricted to the hardware in the lab.)

5.c Avoiding collisions. There are two main sequences of events leading to a collision.

1. Error in time measurements cause the distance between to two trains to vary randomly. The goal that you set for average distance is less than the random variation of distance, so the trains collide periodically.

   (i) In practice, how would you determine the minimum collision-free distance between trains for this sequence of events.

2. The algorithm you choose to adjust the speed is unstable. At first the distance between the trains varies a little, but with time it increases without bound until the two trains collide so hard they fall off the track.

   (i) Make an estimate of the random variation in the first sequence, using typical numbers.

   (ii) Consider how you might estimate the limit beyond which catastrophic oscillation occurs in the second case. Could your software detect that a growing oscillation was occurring in real-time and respond fast enough to avert disaster?

Trying out your algorithms by simulating them using made up data is an effective way of exploring problems like instabilities in algorithms. (Of course, you need realistic input to put into the simulation, which requires good calibration. Groan.)

* It happens like this. When the train falls behind it is given too much acceleration. Because sampling is discrete it gets too close resulting in a larger deceleration, and so on.