IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2011

EEE/ISE PART III/IV: MEng, BEng and ACGI

EMBEDDED SYSTEMS

Thursday, 19 May 10:00 am

Time allowed: 1:30 hours

There are THREE questions on this paper.

Answer TWO questions.

All questions carry equal marks.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : T.J.W. Clarke
Second Marker(s) : Y.K. Demiris

Special Information for Invigilators: none.

Information for Candidates

The Rate Monotonic Analysis condition for a set of N jobs where the ith job has CPU utilisation U_i is:

$$\sum_{1}^{N} U_i < N\left(2^{1/N} - 1\right)$$

The Questions

1.

a) Two engineers are arguing about how to design embedded systems software:

A: "I design all my real-time systems to work correctly independent of scheduling."

B: "I would never be able to do that because my systems have hard deadlines."

Discuss the merits of these two views: is B necessarily correct?

[5]

b) For each case below describe why the specified change could make a non-working real-time system work correctly, illustrating your argument with an example.

(i) Adding an interrupt lock critical section

(ii) Removing an interrupt lock critical section

[5]

c) *Figure 1* shows a list of RTOS features available in an RTOS. Note that critical sections around blocking RTOS API calls prevent interrupts only for the time in which the calling task is active, not while it is blocked.

Provide pseudocode to show, using these features, how you would implement either Priority Inheritance Protocol or Ceiling Priority Protocol to prevent priority inversion.

[10]

Dynamic task priorities
Binary semaphores
Critical sections allowed around all RTOS API calls

Figure 1

2.

a) An embedded system with a 20MHz CPU uses interrupts *A*, *B*, *C*, *D*, as detailed in *Figure 2.1*. State constraints on the priorities you would give these interrupts in a 16 level priority-based system, together with reasons for your choice.

[4]

b) Figure 2.1 displays the CPU time for the ISR computation associated with each job, excluding entry and exit overheads. It is known that 16 registers must be saved & restored on interrupt entry and exit. Register push and pop instructions take 1 and 2 cycles respectively. Stating any assumptions you make, estimate the minimum & maximum latency of each interrupt in μs under priority scheduling as in part a).

[10]

c) Determine the minimum clock speed required for the CPU in this system to meet deadlines for all tasks.

[6]

Job	CPU time / cycles	Period / μs
A	100	25
B	200	40
C	300	150
D	400	100

Figure 2.1

3.

a) *Figure 3.1* shows a real-time system with tasks A, B, C, D analysed as jobs, and semaphores P, Q, R. The job period (P) and job length (C) are given for each task. Before completing its job each task waits on each semaphore at most the number of times indicated by the number in the task row and semaphore column. No semaphore sections are nested. Assume that the extended RMA model may be applied and calculate the condition on job lengths necessary for all deadlines to be met, assuming that the maximum CPU time for any task holding the token for semaphore X is T_X. You may ignore task switching overhead.

[10]

b) Determine which semaphores in this system result in possible priority inversion, and what is the maximum extra system CPU utilisation due to the priority inversion. Therefore calculate a new condition for meeting deadlines if these semaphores implement Priority Inheritance Protocol.

[5]

c) Suppose that the semaphore critical sections used in this system can be nested. Explain why the system will not necessarily meet deadlines for any job lengths. What is this situation called?

[5]

	P	*Q*	*R*	Job period (P) /µs	Job length (C) /µs
A	1	0	0	1	*a*
B	0	1	0	2	*b*
C	0	1	2	4	*c*
D	2	0	1	16	*d*

Figure 3.1

RR 4-52

Answer to Question 1

1.

a)

- Systems should be designed to be functionally correct independent of scheduling, this is one interpretation of what A is saying.
- Systems with hard deadlines can best be implemented using priority scheduling. In principle any scheduling order would be OK if CPU is fast enough, so A correct. In practice coexistence of fast & slow deadlines makes that impossible, B correct

[6]

b) (i) Section around two read/write operations on shared data structure that need to be atomic

(ii) If section removed was longer than any other it will increase interrupt latency. Reduced latency after removal will allow interuptdealine to be met. Note could maybe be replaced by scheduling lock section to preserve atomicity at task level.

[4]

c)

To implement PCP use semaphores, add code to modify task priorities. Protect with critical section. Could implement PIP similarly but this is more complex requiring tasks to change priority of other tasks.

```
CPP_sema_wait( sema)
{
ENTER_CRITICAL_SECTION()
sema_wait(sema)
push current priority onto task-specific stack
current_task.prio = sema.ceiling_prio
END_CRITICAL_SECTION()
}

CPP_sema_signal(sema)
{
sema_signal(sema)
current_task_prio = pop priority from task-specific stack
}
```

[10]

Answer to Question 2

a) A > B > D > C

[4]

b) Assume interrupt entry & return takes 2 cycles (could be anything from 0 to 10). Assume (initially) all latencies are less than 25us, so at most one interrupt at each level can contribute to latency. Then:

E = (2 + 16)/20=18/20=0.9

R = (2+32)/20= 34/20=1.7

A: max latency = E=0.9

B: max latency = 2E+R+5=8.5us

D: max latency = 3E+2R+5+10=20.2us

C: max latency = 4E+3R+5+10+20=43.7us (initial assumption)

This time (C) allows 2 A interrupts contrary to assumption, so need to add 5us

D: max latency = 48.7us

Min latency in all cases is E = 0.9us.

[10]

c) utilisation = (5+2.6)/25+(10+2.6)/40+(15+2.6)/150 +(20+2.6)/100=0.9506

RMA limit is 4*(2^0.25-1)=0.757 => CPU speed needed = 25.1MHz

[6]

Answer to Question 3

3.

a) RMA limit is 0.743.

To use extended RMA we calculate blocking of higher priority tasks by lower due to semaphore.

D: none

C:blocked by D, once only, T_R

B: blocked by C, T_Q

A: blocked by D,T_P, due to PI can also be blocked by B & C (b+c)

so total is a+TP+b+c +(b+TQ)/2+(c+TR)/4+d/16 =

$a+1.5b + 1.25c+d/16+ T_P+0.5T_Q+0.25T_R < 0.743$

[10]

b) P allows PI. Max extra utilisation is b+c. If PIP, then this can't happen, so new limit is a+0.5b+0.25c=d/16+TP+0.5TQ+0.25TR<0.743

[6]

c) If semaphore sections can be nested then deadlock could result since (A,D), (D,C), (C,A) make a dependency cycle on P,R,Q respectively. In this case regrdless of job lengths no deadlines can be met.

[4]