

(Till tentamensvakten: engelsk information behövs)

Exam

Embedded Systems I, DVA454
Västerås, 2018-11-07

Teachers: Saad Mubeen, tel: 021-103191
Adnan Causevic, tel: 021-107059
Guillermo Rodriguez-Navas, tel: 021-101356

Exam duration: 14:10 – 18:30

Help allowed: calculator, language dictionary, ruler

Points: 90 p + extra lab points

| Grading: | Swedish grades: | ECTS grades: |
|----------|-----------------|-----------------|
| | 0 – 54 → failed | 0 – 54 → failed |
| | 55 – 76 p → 3 | 55 – 65 → D |
| | 77 – 90 p → 4 | 66 – 79 → C |
| | 91 – 100 p → 5 | 80 – 90 → B |
| | | 91 – 100 → A |

Instructions:

- Answers should be written in English.
- Short and precise answers are preferred. Do not write more than necessary.
- If some assumptions are missing, or if you think the assumptions are unclear, write down what do you assume to solve the problem.
- Write clearly. If I cannot read it, it is wrong.

Good luck!!

Assignment 1: (14 points)

- a) Is this sentence True or False? Explain why. "The processes to be controlled in a chemical plant are slow. For this reason, chemical plants do not need embedded systems implementing real-time control". (4p)
 - b) Comment on the following statement: "For implementation of embedded systems, ASICs should *always* be preferred to FPGAs because ASICs provide better performance and consume less energy". (4p)
 - c) The code generated after compilation and linking is called the "relocatable" code. Can this code be executed? If not, what is needed in order to make it executable? (6p)
-

Assignment 2: (24 points)

- a) What type of circuit protection is provided by capacitors? Give one example and explain the physical principle behind it. (8p)
 - b) Explain the concept of the "Open collector"/"Open drain" mode to drive a digital output. Describe its role in the WIRED AND configuration and its working principle in the TWI communication bus. (12p)
 - c) Arrange in the correct order the following actions implemented by an Interrupt Service Routine (ISR). (4p)
 - I. restore CPU context
 - II. save CPU context
 - III. acknowledge interrupt
 - IV. handle the interrupt
-

Assignment 3: (12 points)

Imagine that you work in a company as a software developer for resource-constrained embedded systems.

- a) Briefly explain one resource optimization technique that you would use during the development of software for these systems. (4p)

Now imagine that you are asked to act as a "Test Manager" for the next project you do with your colleagues.



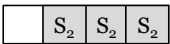
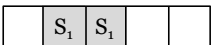
- b) You are asked to explain to your colleagues the connection between Continuous Integration (CI) and testing. Try to elaborate your answer. (4p)

Now imagine that you have been promoted in the same company to lead a team of software designers and integration testers.

- c) Which one of the following strategies would you recommend to the designers of the software modules so that it is easy for the testers to perform the integration testing? Explain and motivate your answer by an example. (4p)
- High coupling and high cohesion
 - High coupling and low cohesion
 - Low coupling and high cohesion
 - Low coupling and low cohesion

Assignment 4: (28 points)

Consider a real-time task set consisting of four tasks, A, B, C, D. The tasks A, C and D share two resources protected by semaphores S_1 and S_2 . Note that task B does not use any shared resource. The tasks have different priorities and they are released for execution at different release times (see table below). The tasks use their semaphores as illustrated in the column "Execution sequence" below (clock ticks are counted relative to the start of the system). The execution times of tasks are as follows: A = 5 ticks, B = 8 ticks, C = 4 ticks and D = 5 ticks as illustrated in the table below (in the "Execution sequence" column). The deadline of each task is relative to its release time. For example, the deadline of Task A is 10, which is relative to its release time 4 (see the table below). This means that relative to time 0, the deadline of Task A is 14. Similarly, the deadline of Task B is 16, which is relative to its release time 6. This means that relative to time 0, the deadline of Task B is 22.

| Task | Priority | Release Time | Deadline (relative to the release time) | Execution sequence |
|------|----------------|--------------|--|--|
| A | 4 (Highest) | 4 | 10 |  |
| B | 3 | 6 | 16 |  |
| C | 2 | 2 | 20 |  |
| D | 1 (Lowest) | 0 | 22 |  |
| | | | | Clock Tick 0 1 2 3 4 5 6 7 8 |

For example, we can see in the table that task A has the highest priority, it is released at time $t = 4$, and, once released, it will execute like described below:

- tick 4+0*: tries to execute one clock tick without any semaphores.
- tick 4+1*: tries to lock semaphore S_2 , and if ok, it enters its critical section with S_2 .
- tick 4+2*: tries to execute one clock tick without any semaphores.
- tick 4+3*: tries to lock semaphore S_1 , and if ok, it enters its critical section with S_1 .
- tick 4+4*: tries to execute one clock tick without any semaphores.

The same reasoning applies to all other tasks.

Note that the execution scenarios for the tasks will be equal to the ones illustrated in the table above *only under the assumption* that the required semaphores are *free* when requested by a task, and the task is not pre-empted by a higher-priority task. However, from the release times above we can see that the tasks will interfere with each other. Besides, the semaphores will not be always available when requested by the tasks. Assume the release times of the tasks, their priorities and the execution sequences from the above table.

Part 1: Without using any synchronization protocol

- a) Is the task set schedulable WITHOUT using any synchronization protocol? Draw the actual execution trace for the given task set WITHOUT using any synchronization protocol. You should run your trace from time $t=0$ until all of the tasks have completely executed once their execution sequence. (4p)
- b) Does priority inversion occur in Part 1(a)? If yes, then clearly identify the time interval(s) where the priority inversion occurs. If not, then explain why not. (2p)
- c) Does chained blocking occur in Part 1(a)? If yes, then clearly identify the time interval(s) where the chained blocking occurs. If not, then explain why not. (2p)

Part 2: Use the Priority Inheritance Protocol

- a) Is the task set schedulable if the *Priority Inheritance Protocol (PIP)* is used? If not, then why not? Draw the actual execution trace from time $t=0$ until all of the tasks have completely executed once their execution sequence. (6p)
- b) Does priority inversion occur in Part 2(a)? If yes, then clearly identify the time interval(s) where the priority inversion occurs. If not, then explain why not. (2p)
- c) Does chained blocking occur in Part 2(a)? If yes, then clearly identify the time interval(s) where the chained blocking occurs. If not, then explain why not. (2p)

Part 3: Use the Priority Ceiling Protocol

- a) Is the task set schedulable if the *Priority Ceiling Protocol (PCP)* is used? If not, then why not? Draw the actual execution trace from time $t=0$ until all of the tasks have completely executed once their execution sequence. (6p)
- b) Does priority inversion occur in Part 3(a)? If yes, then clearly identify the time interval(s) where the priority inversion occurs. If not, then explain why not. (2p)
- c) Does chained blocking occur in Part 3(a)? If yes, then clearly identify the time interval(s) where the chained blocking occurs. If not, then explain why not. (2p)

Important Note regarding Parts 1(b), 1(c), 2(b), 2(c), 3(b) and 3(c): Any "Yes or No" answer without identification of the interval(s) or justification of the answer will not be awarded any point.

Assignment 5: (12 points)

Assume three periodic tasks τ_1, τ_2 and τ_3 that communicate among each other by sending messages among their instances. The following is given:

Task τ_1 :

- Has execution time of 2 ms and period of 12 ms.
- Sends 2 messages to a message queue MSGQ during each instance (job).
- All the messages are sent at the end of execution of each job.

Task τ_2 :

- Has execution time of 1 ms and a period of 6 ms.
- Sends 2 messages in the message queue during each instance (job).
- All the messages are sent at the end of execution of each job.

Task τ_3 :

- Has execution time of 2 ms and a period of 4 ms.
- Receives 2 messages from the message queue during each instance (job).
- All the messages are received at the end of execution of each job.

MSGQ:

- The queue contains the copy of the messages (not pointers).
- Has First In First Out (FIFO) order for inserting the messages.
- When a task reads a message from the message queue, the message is removed from the queue.

Question:

Assume that the tasks are scheduled using the **Earliest Deadline First** algorithm. What is the minimum possible size of the message queue (counted in number of messages) such that we are able to guarantee there will always be enough space in the queue for τ_1 and τ_2 to insert their messages? Motivate your answer by drawing an execution trace up to one hyper period and showing the number of messages in the queue after the execution of each task instance. (12p)

Assignment 6: (extra lab points)

You do not need to do anything here. This is for the extra points earned at the labs. Your extra lab points will be automatically added to your total exam score.
