(Till tentamensvakten: engelsk information behövs)

Exam

Embedded Systems I, DVA454
Västerås, 2019-11-06

Teachers:	Saad Mubeen, tel: 021-103191 Mohammad Ashjaei, tel: 021-151772 Adnan Causevic, tel: 021-107059
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: (10 points)

a) What is the difference between a *native* compiler and a *cross* compiler? (4p)

b) Explain what is a "relocatable" code. Is this code executable? If not, what process is needed to make it executable? (2p+2p+2p)

Assignment 2: (10 points)

a) What is an advantage and a disadvantage of using ASIC instead of FPGA? Explain briefly. (4p)

b) Draw a simple circuit to use an LED. Calculate and set necessary values for different components in the circuit, e.g., resistor. Assume a DC voltage source. (2p+4p)

Assignment 3: (12 points)

a) Explain a scenario in which an ISR will be delayed unintentionally? Draw a trace to motivate your answer. (4p+4p)

b) What is a watchdog timer? Describe an example in which a watchdog timer is helpful? (2p+2p)

Assignment 4: (12 points)

a) What is a re-entrant code? (2p)

b) Write a small example code that is not re-entrant? Motivate why is this code not re-entrant. (4p+2p)

c) Give suggestions to make the example code in (b) re-entrant. (4p)

Assignment 5: (12 points)

a) Briefly explain one resource optimization technique that you would use during the software development of resource-constrained embedded systems. Motivate your answer with an example (6p)

b) What is the difference between simulation and emulation from the testing perspective? Explain your answer. (6p)

Assignment 6: (16 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 4 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 3 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.

Questions:

a) Assume that the tasks are scheduled using the **Rate Monotonic** 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 τ1 and τ2 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. (8p)

b) 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 τ1 and τ2 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. (8p)

Assignment 7: (18 points)

Consider a real-time task set consisting of five tasks, A, B, C, D, E that share four resources protected by semaphores S1, S2, S3, S4. The tasks have different priorities and they are released for execution at different release times (see table below). All 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 = 9 ticks, B = 4 ticks, C = 5 ticks, D = 5 ticks and E = 4 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 B is 16, which is relative to its release time 8 (see the table below). This means that relative to time 0, the deadline of Task B is 24.

Task	Priority	Release time	Deadline (relative to the release time)	Execution sequence
A	5 (Highest)	10	11	S1 S2 S3 S4
B	4	8	16	S4 S4
C	3	5	20	S3 S3
D	2	2	25	S2 S2 S2
E	1 (Lowest)	0	27	S1 S1

Clock Tick 0 1 2 3 4 5 6 7 8 9

For example, we can see in the table that task B has the second highest priority, Prio (B) = 4, it is released at time t = 8, and, once released, it will execute like described below:

- *tick 8+0*: tries to execute one clock tick without any semaphores.
- *tick 8+1*: tries to lock semaphore S4, and if ok, it enters its critical section with S4.
- *tick 8+2*: tries to continue its critical execution with S4. It then releases S4 at the end of the tick.
- *tick 8+3*: 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 table above:

a) Is the task set schedulable if the *Priority Inheritance Protocol (PIP)* is used? If not, then why not? Draw the actual execution trace. You should run your trace from time t=0 until all of the tasks have completely executed once their execution sequence. (8p)

b) Does chained blocking occur in (a)? If yes, then identify the time interval(s) where the chained blocking occurs. If not, then why not? (2p)

Note: Any "Yes or No" answer without identification of the interval(s) or justification of the answer will not be awarded any point.

c) Is the task set schedulable if the *Priority Ceiling Protocol (PCP)* is used? If not, then why not? Draw the actual execution trace. You should run your trace from time t=0 until all of the tasks have completely executed once their execution sequence. (8p)

Assignment 8: (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.