MASTER SETI / MASTER CPS / Option SE
Critical Real-Time Systems

Points for grading may vary, **no documents allowed (closed book)**.

The clarity and style of your explanations will be considered during grading.
Only readable, precise and sufficiently explained answers will be considered.

Part 6 is exclusive for SETI and SE students, part 7 is exclusive for CPS students.

1. Code Coverage (3 points)

1.1. Code coverage [course] (1 point)

Indicate what should be done by a test suite to guarantee the coverage of code according to the approaches 1) Statement Coverage, 2) Decision Coverage and 3) Modified Condition / Decision Coverage..

1.2. Applying these approaches SC, DC, MC/DC (2 points)

Consider the code below. Illustrate each of these approaches by providing possible values of a, b, and c for the conditions. Particular attention will be paid to readability and clarity.

if ((a == 0) || ((b != 1) && (c == 2))) {F_1(a, b, c)}; F_2(a, b, c) ;

2. Response time on CAN bus (3 points)

2.1. CAN Bus [course] (1 point)

Recall how collision arbitration works on the CAN bus.

2.2. Message schedulability (1 point)

Recall how the response time for sending a message on a CAN bus is calculated. Give the formula for calculating the response time.

2.3. Applying message schedulability (1 point)

Consider the following set of messages exchanged on a CAN bus. Calculate their response times and determine if this set of messages is schedulable.

Message	Com Time	Period	Emitter
M1	2	10	1
M2	5	20	2
M3	6	40	3

3. Integrated Modular Avionic (3 points)

3.1. Scheduling configuration (2 points)

The table below provides a set of tasks with period, level of criticality, and budget in terms of CPU utilization. Assuming each task is assigned to a unique ARINC653 partition of its criticality level,

determine the “Major Frame”, “Minor Frame” and “Windows Partitions”. By assuming that the most critical partitions are privileged in the scheduling, calculate a possible timing diagram.

Task	Period	Criticality	Budget
T1	100 ms	A	5.00%
T2	200 ms	A	20.00%
T3	100 ms	B	20.00%
T4	200 ms	B	30.00%
T5	100 ms	C	25.00%

3.2. Communication time (1 point)

It is now assumed that at each of its executions, the task T1 sends a datum d to task T5. It is assumed that the inter-partition communications ports are flushed every Minor Frames (MIF). We consider that the emission date of d corresponds to the end date of the “Partition Window” in which T1 is running, and the reception date of d corresponds to the start date of the “Partition Window” in which runs T5. By reusing the scheduling configuration you established in the previous question, after how long does task T5 receive the last value produced by task T1? Justify your answer and represent this delay on the chronogram obtained in the previous question.

4. Global vs partitioned scheduling [problem] (5 points)

In this problem, we study the scheduling of a video surveillance application consisting of a set of tasks described below.

- The **Ta1** task takes care of the audio input device and manages the acquisition of audio samples. **Ta1 provides the inputs of the task Ta2 as an output**.
- The **Ta2** task manages the audio output device (speakers) and presents the audio stream on a supervision console. **Ta2 takes the outputs of Ta1 as input**.
- The **Tv1** and **Tv2** tasks perform similar functions for the video stream.

An engineer has to validate the temporal behaviour of this task set. It is further assumed that these are defined by the parameters below:

Task	**Budget**	**Period**
Ta1	5	10
Ta2	5	10
Tv1	10	20
Tv2	10	20

The deadlines are equal to the periods (tasks with implicit deadlines). The date of first activation of all tasks is the same and is zero (synchronous tasks).

Assume a **preemptive fixed priority scheduler** having 100 priority levels (from 0 to 99, the priority level 99 being the highest priority level).

4.1. Priorities to make tasks independent (0,5 point)

The tasks are not independent (see description above). The task **Ta2** (respectively **Tv2**) must be activated on termination of the task **Ta1** (respectively **Tv1**).

From **Ta1** and **Ta2** (respectively **Tv1** and **Tv2**), which task must have the highest priority in order to ensure the precedence constraints expressed above.

Once these constraints on priorities have been expressed, tasks Ta1 and Ta2 (resp. Tv1 and Tv2) will be considered as independent. They are activated at the beginning of their period (a_i=0) but their assignment of priority must guarantee that the precedences are respected.

4.2.Partitioned scheduling (independent tasks) (1 point)

The execution platform consists of two identical processors. As a first approach, we want to apply a partitioned scheduling: the tasks **Ta1** and **Ta2** are placed on a first processor and the other tasks on the second. We use the properties on the priorities expressed in **question 4.1**. Calculate the scheduling over the hyper-period. Is the system schedulable?

4.3.Global scheduling (independent tasks)(0.5 point)

In a second approach, a preemptive global scheduling with fixed priority is used to execute these tasks on the 2 processors, the allocation of priorities being to be determined. It is assumed that the migration of tasks can take place at any time. In the following, it is assumed that this scheduling makes it possible to correctly execute the set of tasks **Ta1**, **Ta2**, **Tv1** and **Tv2** while ensuring the properties on the priorities expressed in **question 4.1** in order to make these tasks independent.

Given the capabilities of **Tv1** and **Tv2** demonstrate that they necessarily consume a full processor over the entire hyper-period.

4.4. Property on priorities of Tv1 and Ta1 (0.5 point)

Prove at time t = 0, **Tv1** and **Ta1** must have the 2 highest priorities for the system to run properly.

4.5.Property on priorities of Tv2 and Ta2 (0.5 point)

Prove that one must have priority(**Tv2**) < priority(**Ta2**) for the system to run correctly at t = 5.

4.6.Opposite property on priorities of Tv2 and Ta2 (0.5 point)

Prove that one must have priority(**Tv2**) > priority(**Ta2**) for the system to run correctly after t = 10.

4.7.Comparison of approaches on a particular scenario (0.5 point)

For this particular scenario, which approach (global or partitioned) would you recommend? Why?

5. WCET Analysis (3 points)

5.1. Static Analysis (1.5 points)

a) Explain the three components that constitutes the “Constant Propagation” analysis covered in the course: the abstract domain, the transfer function, and the meet/join operator. Which mathematical structure is used to represent the domain? Explain the link between this mathematical structure and the meet/join operator.

b) What is a Basic Block? Which role do basic blocks play in control flow graphs (CFG)?

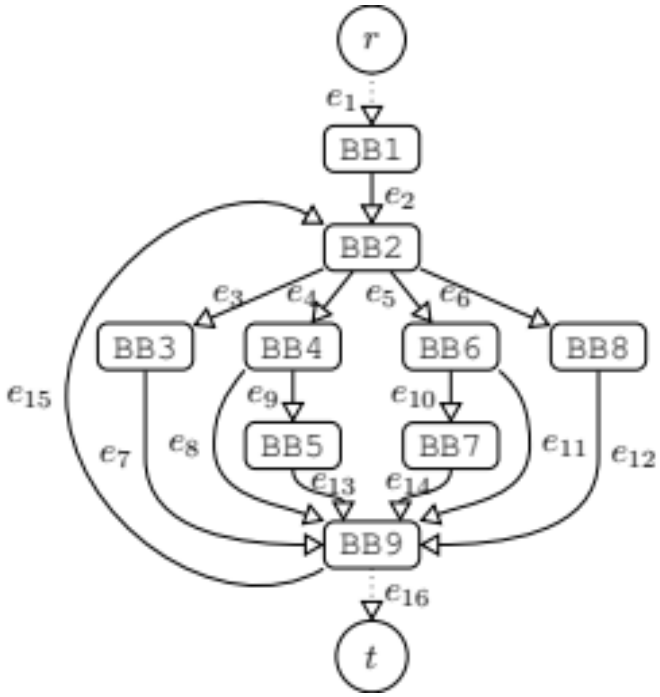
5.2. Implicit Path Enumeration (IPET) (1.5 points)

For the following questions consider the C code and control-flow graph (CFG) shown on the next page. The code implements a state machine that handles the reception of data in bursts (state RECEIVE) and the processing of that data (state PROCESS).

a) The number of times that this state machine executes the basic block **BB6** (state RECEIVE) is executed can be bounded by NUM_BLOCKS and BURST_SIZE. Give a linear equation for the Implicit Path Enumeration Technique (IPET) that allows to bound the number of executions of this block. You may use symbolic names from the code, names of basic blocks, as well as names of edges from the control-flow graph in your formula. Justify your solution in 2-3 sentences.

b) We now wish to add additional equations to bound the number of executions of **BB2**. If this is possible, provide a solution and explain it. Otherwise, explain why it is not possible to such IPET equations.

```
// BB1
b = 0;
state = INIT;
do
{
 // BB2
 switch (state)
 {
  case PROCESS:
   // BB3
   b++;
    state = IDLE;
   break;
  case IDLE:
   // BB4
   if (request)
   {
    // BB5
    state = RECEIVE;
    transferred = 0
   }
   break;
  case RECEIVE:
   // BB6
   transferred++;
   if (transferred == BURST_SIZE)
   {
    // BB7
    state = PROCESS;
   }
   break;
  case INIT:
   // BB8
   state = IDLE;
   break;
 }
 // BB9
} while (b != NUM_BLOCKS);
```



6. WCET and Cache Analysis (3 points) [SETI/SE301a]

6.1. Cache/Memory

The following questions concern the analysis of **instruction caches**, i.e., the classification of memory accesses based on the memory blocks that are accessed during the execution of the basic blocks. Consider a **fully-associative cache** with a size of **4 blocks** and the **least-recently used** replacement policy (LRU). The following figure shows the association between memory blocks and the basic blocks of the program from the previous page:

6.2. Hit/Miss Classification (1 point)

a) Explain the **MUST** analysis and which kind of classification it may provide for accesses to memory blocks. What is the link between the age af a memory block during the actual execution of the program and the information computed by the analysis?

b) Indicate a basic block of the program for which the **MUST** analysis is able to provide a classification – if such a block exists. Justify your answer.

6.3. Persistence (2 points)

a) Hit/Miss classification may sometimes give pessimistic results for certain memory accesses, notably in loops. Explain how **persistence** is able to overcome this pessimism?

b) Verify whether the memory block m_1 accessed during the execution of **BB2** is persistent. Based on your analysis answer to one of the following two questions:

- m_1 is persistent:
  Does this imply that all memory blocks accessed inside the loop are also persistent?
- m_1 is not persistent:
  Does this imply that all the memory blocks accessed in the loop are non-persistent?

7. Multi-processors scheduling (3 points) [CPS]

In this exercise, we seek to schedule the following system of periodic synchronous tasks with implicit deadlines on a two-core processor:

Tâches	*Budget*	*Période*
T1	2	8
T2	2	10
T3	36	40

7.1. Global-EDF Scheduling [course] (0.5 points)

Recall how Global-EDF works.

7.2. Applying Global-EDF to a given task set (1 point)

Is this task set schedulable with Global-EDF ?

7.3. Global-EDF applied to a less loaded task set (1 point)

The period of T1 is increased from 8 to 10. Is this system schedulable using Global-EDF?

7.4. Classic phenomenon on multi-processors scheduling (1 point)

What classic phenomenon of multi-processor schedulers do these examples illustrate?