Examen
Durée : 2h00
Barême indicatif. Les notes de cours ne sont pas autorisées.
Il sera tenu compte de la présentation et de la clarté dans la rédaction.

Les parties 1-2 et 3-4 doivent être traitées sur des copies séparées. La partie 2 concerne les étudiants de SE301 et COMASIC. La partie 4 concerne les étudiants de SETI.

1 Systèmes temps réel critiques

Exercice 1 : Ordonnancement global vs partitionné (5 points)

Dans cet exercice, nous étudions l'ordonnancement d'une application de vidéosurveillance composée d'un ensemble de tâches décrites ci-dessous.

— La tâche **Ta1** s'occupe du périphérique d'entrée audio et gère l'acquisition des échantillons audio. **Ta1 fournit en sortie les entrées de la tâche Ta2**.
— La tâche **Ta2** gère le périphérique de sortie audio (haut-parleurs) et présente le flux audio sur une console de supervision. **Ta2 prend en entrée les sorties de Ta1**.
— Les tâches **Tv1** et **Tv2** assurent des fonctions similaires pour le flux vidéo.

Un ingénieur souhaite valider le comportement temporel de ce jeu de tâches. On suppose, en outre que celles-ci sont définies par les paramètres ci-dessous :

Tâches	Capacité	Période
Ta1	9	20
Ta2	9	20
Tv1	20	40
Tv2	20	40

On modélise ce système sous forme de tâches synchrones à échéances implicites. On suppose un ordonnanceur préemptif à priorité fixe disposant de 100 niveaux de priorité (de 0 à 99, le niveau de priorité 99 étant le niveau de priorité le plus élevé.

La plate-forme d'exécution est constituée d'un processeur composé de deux coeurs identiques.

On souhaite modéliser le problème sous forme de tâches indépendantes en imposant des priorités telles que les contraintes de précédences soient respectées.

(a) (1 point) **Priorités entre Ta1 et Ta2**

D'après la description du problème, la tâche Ta2 (respectivement Tv2) doit être activée sur terminaison de la tâche Ta1 (respectivement Tv1).

De Ta1 et Ta2 (respectivement Tv1 et Tv2), quelle tâche doit avoir la plus forte priorité afin d'assurer les contraintes de précédence exprimées ci-dessus ?

(b) (1 point) **Approche partitionnée**

En première approche, on souhaite appliquer un ordonnancement partitionné : les tâches Ta1 et Ta2 sont placées sur un premier coeur et les autres tâches sur le second. On utilise les propriétés sur les priorités exprimées à la question précédente.

Calculer l'ordonnancement sur l'hyper-période. Le système est-il ordonnançable ?

(c) **Approche globale**

En seconde approche, on utilise un ordonnancement global préemptif à priorité fixe pour exécuter ces tâches sur les 2 coeurs. On suppose que la migration des tâches peut intervenir à tout moment.

Dans la suite, on suppose que cet ordonnancement permet d'exécuter correctement l'ensemble de tâches Ta1, Ta2, Tv1 et Tv2 tout en assurant les propriétés sur les priorités exprimées à la question 1.1 afin de rendre ces tâches indépendantes.

i. ($\frac{1}{2}$ point) **Priorités de Ta1 et Tv1**

Démontrer qu'à l'instant t=0, Tv1 et Ta1 doivent avoir les 2 priorités les plus fortes pour que le système s'exécute correctement.

ii. (1 point) **Priorités entre Tv2 et Ta2 à t=9**

Démontrer que l'on doit avoir $priority(Tv2) < priority(Ta2)$ pour que le système s'exécute correctement à t=9.

iii. (1 point) **Priorités entre Tv2 et Ta2 à t=20**

Démontrer que l'on doit avoir $priority(Tv2) > priority(Ta2)$ pour que le système s'exécute correctement à t=20.

(d) ($\frac{1}{2}$ point) **Conclusions**

En conclusion, quelle approche (globale ou partitionnée) recommanderiez-vous ? Pourquoi ?

Exercice 2 : Ordonnancement global (3 points)

Dans un système comportant un processeur composé de deux cœurs identiques, les tâches sont exécutées suivant une politique d'ordonnancement Global Deadline Monotonic. Soit l'ensemble de tâches périodiques synchrones suivantes :

Tâche	Capacité	Echéance	Période
T1	1	4	4
T2	3	5	5
T3	8	9	20

(a) ($1\frac{1}{2}$ points) **Premier ordonnancement**

Calculer l'ordonnancement sur l'hyper-période. Toutes les échéances sont-elles respectées ?

(b) ($1\frac{1}{2}$ points) **Second ordonnancement**

Supposons maintenant que la période de T1 soit égale à 5 l'échéance restant à 4. Calculez à nouveau l'ordonnancement sur l'hyper-période. Que constatez-vous maintenant ?

Exercice 3 : Ordonnançabilité avec partage de ressources (3 points)

Dans un système comportant un mono-processeur, les tâches sont exécutées suivant une politique d'ordonnancement Rate Monotonic. Soit l'ensemble de tâches périodiques synchrones suivantes partageant deux ressources R1 et R2. Les colonnes 4 et 5 désignent les temps d'exclusion mutuelle sur les ressources R1 et R2. Lorsqu'il vaut 0, la ressource n'est pas utilisée.

Tâche	Capacité	Période	Temps de Mutex sur R1	Temps de Mutex sur R2
T1	2	8	1	0
T2	3	10	2	1
T3	3	20	0	1
T4	7	40	2	1

Nous utilisons le protocol Priority Inheritance Protocol. Une tâche T peut bloquer au plus une fois sur chacune des ressources qu'elle utilise ; une tâche moins prioritaire ne peut bloquer la tâche T qu'une seule fois ; une tâche moins prioritaire peut bloquer la tâche T indirectement sans partager de ressource lors d'un héritage de priorité.

(a) ($1\frac{1}{2}$ points) **Calcul du temps de blocage**
A partir des remarques précédentes, calculez le pire temps de blocage des tâches T1 à T4 en expliquant votre raisonnement.

(b) ($1\frac{1}{2}$ points) **Pire temps de réponse**
A partir des résultats précédents, calculez le pire temps de réponse des tâches T1 à T4 en expliquant votre raisonnement. Précisez également si les tâches sont ordonnançables.

Exercice 4 : Couverture de code (3 points)

On considère le code suivant. Indiquez ce que doit faire un banc de tests qui cherche à garantir la couverture de code selon l'approche Modified Condition/Decision Coverage. Puis illustrez en fournissant des valeurs possibles de a, b, c et d pour les tests à fournir. Une attention particulière sera accordée aux explications.

```
if (((a == 1) && (b < 2)) || ((c != 3) && (d == 4)))
   F1(a, b, c, d);
 F2(a, b, c, d);
```

Total des points de la section : 14 points

2 Protocol CAN (COMASIC et SE301 UNIQUEMENT)

Exercice 5 : Bus CAN (2 points)

Expliquez pourquoi le protocole d'accès au bus CAN est qualifié de bitwise arbitration. Justifiez sur un exemple mettant en jeu trois sites. Expliquez comment il est possible de calculer le pire temps de communication d'un message en modélisant le système sous forme d'un problème d'ordonnancement. On précisera l'algorithme d'ordonnancement à utiliser pour CAN.

Total des points de la section : 2 points

3 Pire temps d'exécution

Exercice 6 : Analyse Statique (1 points)

Quel est l'avantage de l'analyse statique dans le contexte de l'analyse du pire temps d'exécution par rapport à de simples mesures du temps d'exécution sur une plateforme matérielle ?

Exercice 7 : Analyse des Ranges (1 points)

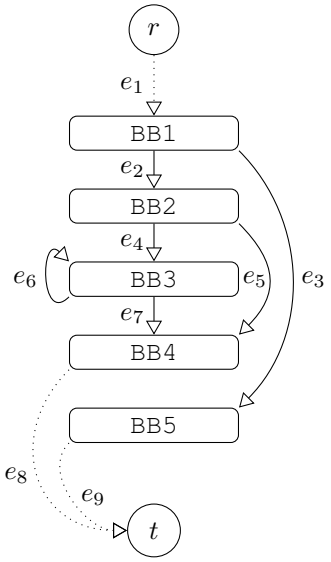
En cours, vous avec vu l'analyse des ranges (Value Range Analysis) et son application dans le contexte de l'analyse du pire temps d'exécution. Quel type d'information est obtenu par l'analyse des ranges ? Quel type de comportement l'information obtenue par l'analyse permet-elle de borner ?

Exercice 8 : Équations d'IPET (2 points)

La figure suivante vous donne le graphe de flot de contrôle (CFG) de la fonction `count_str`. Expliquez le rôle des équations ci-dessous dans l'approche IPET. Pour chaque équation donnez une explication des termes mathématiques (e.g., $120 * e_4$).

Attention : Il y a potentiellement plusieurs appels à la fonction. C'est à dire le flux pour e_1 est potentiellement plus large que 1.

— $e_1 = e_2 + e_3$
— $e_5 = 0$
— $e_4 + e_6 \leq 120 * e_4$



Total des points de la section : 4 points

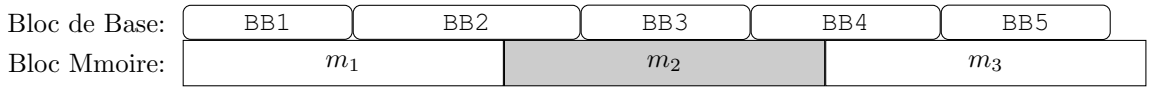
4 Pire temps d'exécution (SETI UNIQUEMENT)

Exercice 9 : Bloc Mémoire ($\frac{1}{2}$ points)

Expliquez le terme *bloc mémoire* (Memory Block) et son utilisation dans l'analyse de cache.

Exercice 10 : Classification des Accès au Cache ($1\frac{1}{2}$ points)

Le figure ci-dessous indique le placement des blocs de base (Basic Block) de la fonction `count str`-(voir le CFG de la question 3). Notamment, durant l'ex´ecution de `BB3`, le bloc m´emoire (memory block) $m2$ est acc´ede´ et durant l'ex´ecution de `BB4` les deux blocs m´emoires $m2$ et $m3$ sont acc´ed´es. En supposant un cache avec une associativite´ de 4, les analyses **MUST** et **MAY** indiquent un ˆage de 0 et 0 pour $m2$ au d´ebut de `BB3` respectivement. Pour $m3$ au d´ebut de `BB4` les deux analyses indiquent un age de 4 et 4 respectivement. Quelle est la classification finale de ces deux acc`es (always hit, always miss, not classified) ? Justifiez votre r´eponse en vous appuyant sur le CFG de la fonction.



Total des points de la section (SETI UNIQUEMENT) : 2 points