Date : 10 mars 2011

École Polytechnique de Montréal

INF3610 : Systèmes embarqués

Questionnaire du QUIZ : Hiver 2011

PRENEZ NOTE DES DIRECTIVES SUIVANTES :

- Documentation non permise
- Durée de l'examen : 120 minutes i.e. 10h00 à 12h00
- Ne pas écrire en rouge
- L'examen comporte 4 questions pour un total de 20 points et il compte pour 20% de la note finale. La distribution des numéros est la suivante :

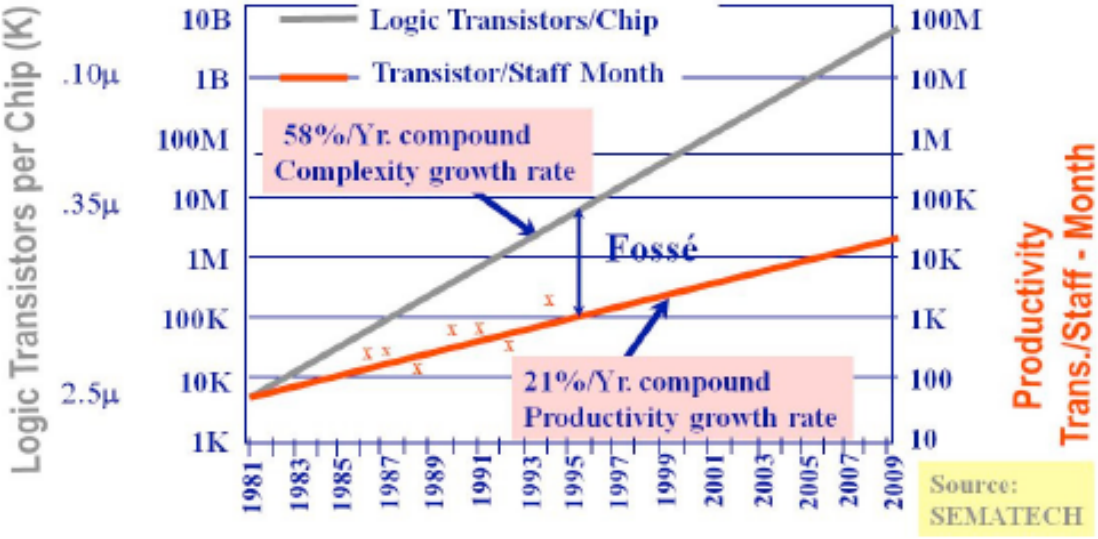
Question 1	4.5 points
Question 2	4.5 points
Question 3	6.5 points
Question 4	4.5 points

- L'utilisation d'une calculatrice non programmable est permise

QUESTION #1 (4.5 points) Notions de base

a) **(2 point)** Expliquez les défis actuels pour la conception des systèmes sur puce. Donnez les solutions possibles pour surmonter ces défis.

Le défi majeur est donné par le fossé de production



Solutions :
1. Il faut faire de la réutilisation, autant au niveau logiciel que matériel
2. Il faut travailler à haut niveau d'abstraction (ESL pour Electronic System Level)
3. Du point de vue architectural, il faut exploiter le parallélisme et l'hétérogénéité

b) (1.5 points) Expliquez pourquoi la combinaison du logiciel et du matériel peut-être nécessaire dans le flot de conception des systèmes temps réel. Donnez un exemple d'application temps réel qui peut exploiter cette combinaison des modèles.

Le logiciel est utilisé pour la flexibilité et le matériel pour la performance.
Exemple : dans une caméra numérique – le traitement d'image peut-être fait en matériel et l'interfaçage avec l'utilisateur en logiciel.

c) **(1 point)** Associez-vous les systèmes d'exploitation temps-réel aux systèmes dominés par le flot de contrôle ou aux systèmes orientées dominés par le flot de données ? Justifiez votre réponse.

Les RTOS sont associés avec les systèmes dominés par le flot de contrôle. Ceci est expliqué par les caractéristiques de ces systèmes :
– Sujet à des contraintes temporelles (débit)
• On doit avoir des processeurs/contrôleurs dont le comportement est facilement prévisible (Exemple: pas de prédiction de branchement)
– Beaucoup de FSM (processus) se partagent le CPU

• Changement de contexte rapide (< 1µs)
– Très peu de données associées aux états d'une FSM
• Beaucoup de petites banques de registres (accès aux données très rapide)
– Doit supporter la préemption
• Interruption de l'exécution d'un processus pour en exécuter un autre plus priotaire

QUESTION #2 (4.5 points) Logiciel embarqué. Concepts Temps Réel

a) (**1.5 point)** Donnez la définition des termes : latence d'interruption, ISR et préemption.

Latence d'interruption : le temps max. pendant lequel les interruptions sont désactivées + le temps nécessaire pour commencer la première instruction de l'ISR

ISR (Interrupt Service Routine) : un code qui traite l'interruption. Son appel par l'OS ou un pilote est déclanché par l'arrivé de l'interruption.

Préemption : Interruption de l'exécution d'un processus pour en exécuter un autre plus priotaire

b) **(2 points)** Expliquez le phénomène présenté dans la Figure 1. Donnez aussi une explication pour les actions représentées de 1 à 7.

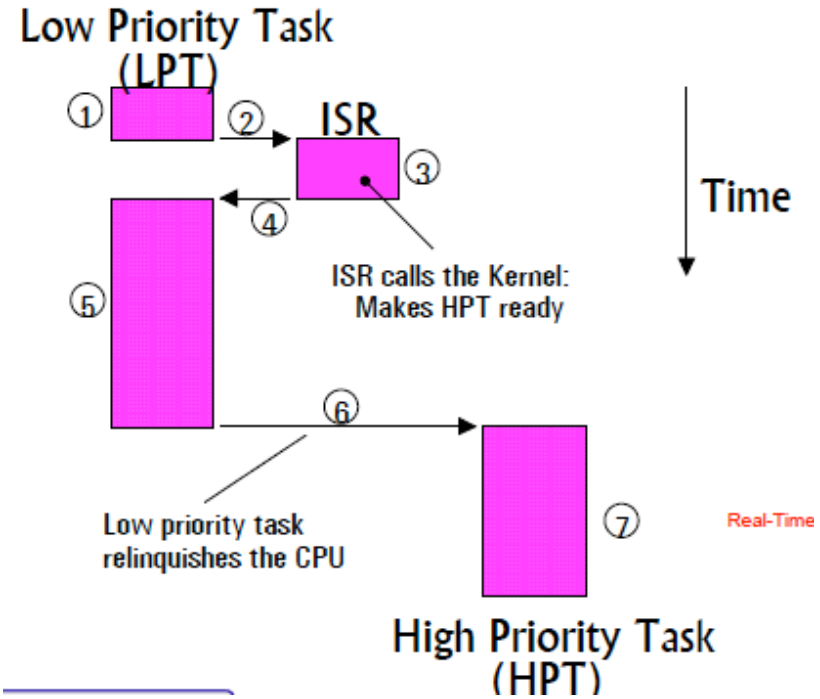


Figure 1

La Figure montre le traitement d'une interruption pour un système non-préemptif

1. Execution de la tâche en cours
2. Arrivée de l'interruption, déclanchement de l'ISR. Ce traitement rend une tâche plus prioritaire prête à s'exécuter
3. Traitement de l'interruption
4. Appel du noyau à la fin de l'ISR
5. Exécution de la tâche intrrompue
6. Changement de contexte vers la tâche plus prioritaire prête pour exécution
7. Exécution de la tâche plus prioritaire

c) **(1 point)** Dérivez le rôle des fonctions d'entrée dans une interruption (ex. OSINTEnter in MicroC) et de sortie d'une interruption (ex. OSIntExit() in MicroC).

OSIntEnter() :
-Disables interrupts to gain exclusive access to the global variable OSIntNesting (current number of interrupt nested
- Increments OSIntNesting
- Enables interrupts

OSIntExit() :
-Disables interrupts
-Decrements OSIntNesting
If OSIntNesting == 0, determines HPT (Highest Priority Task) as the next task to be executed
-Performs a context switch to resume HPT
-Enables interrupts;

QUESTION #3 (6.5 points) MicroC. Caractéristiques, gestion des tâches. Gestion des événements.

a) **(1.5 points)** Répondre par vrai ou par faux aux questions suivantes:

1. (0.5 points) En MicroC, un des paramètres de la fonction de création d'un mutex correspond à un niveau de priorité. Justifiez votre réponse.

Vrai : cette priorité est utilisé pour éviter une éventuelle inversion de priorité

2. (0.5 points) Pour toute application MicroC, le noyau trouve toujours dans le système une tâche prête à s'exécuter. Justifiez votre réponse.

Vrai : il s'agit du rôle de la tâche Idle

3. (0.5 points) Dans l'implantation courante de MicroC, OSEventTable est un champ du TCB (Task Control Block) de chaque tâche. Justifiez votre réponse.

Faux : cette table est associée à un événement mais pas à une tâche

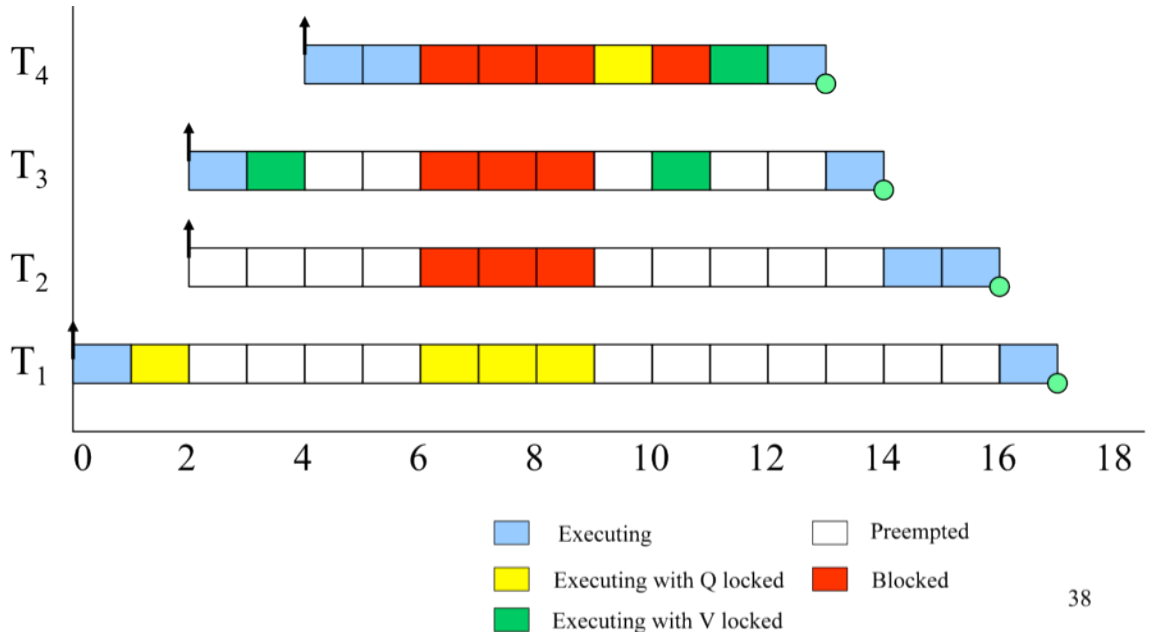
b) **(5 points)** Soit une application de 4 tâches ordonnancée par le RTOS MicroC. Les caractéristiques des quatre tâches sont données à la Table 1.

Tâche	Priorité	Période de départ	Séquence d'exécution
T4	1	4	EEQVE
T3	2	2	EVVE
T2	3	2	EE
T1	4	0	EQQQQE

Table 1

Dans la séquence d'exécution de chaque tâche, chaque lettre correspond à un cycle, la signification étant la suivante:
E – cycle d'exécution normale
V - cycle d'exécution utilisant la ressource partagée V
Q - cycle d'exécution utilisant la ressource partagée Q
Les ressources partagées sont gardées par des mutex.
La période minimale imdique la période (ou cycle) pendant laquelle la tâches est prête à s'exécuter (example : la tâche 4 devient prête à s'exécuter au cycle 4 et pas avant).
Pour les priorités, considérez la convention utilisée en MicroC.

1. (2.5 points) Donnez les traces d'exécution pour les quatre tâches. Attention : Tenez compte que MicroC implante un mécanisme d'inversion de priorités dans les services de gestion des mutex.



2. (2.5 points) Donnez les valeurs des variables OSRdyTbl et OSRdyGrp pendant les cycles d'exécution 4 et 5 (compter les cycles à partir de la valeur 1). Si la variable OSRdyTbl n'est pas la même pour ces deux cycles, montrez comment MicroC a fait la mise à jour de cette variable (au besoin utilisez les annexes).

Cycle 4 : OSRdyTbl [0] = [000011100] OSRdyGrp = [10000001]
Tâche en exécution :

```
Y       =       OSUnMapTbl[OSRdyGrp] ;
X       =       OSUnMapTbl[OSRdyTbl[Y]] ;
prio    =       [Y * 8] + X ;
```

Cycle 5 : OSRdyTbl [0] = [000011110] OSRdyGrp = [10000001]
Tâche en exécution :

```
Y       =       OSUnMapTbl[OSRdyGrp] ;
X       =       OSUnMapTbl[OSRdyTbl[Y]] ;
prio    =       [Y * 8] + X ;
```

Pour mettre à jour OSRdyTbl :

```
OSRdyGrp                |=      ptcb->OSTCBBitY
OSRdyTbl[ptcb->OSTCBY]|=      ptcb->OSTCBBitX;
```

QUESTION #4 (4.5 points) Gestion du temps et gestion de la mémoire.

a) **(3 points)** Dans la tâche la plus prioritaire d'un système, on fait appel à un délai de 14 ms. Sachant que la minuterie (timer) est réglée à une période de 3 ms, quelle sera la précision de ce délai? Expliquez et faites un schéma de l'exécution.
Donnez deux solutions pour améliorer la précision du délai. Pour chaque solution expliquez les avantages et les inconvénients.

No de cycles d'attante : 14/3 = 4.6 => 5 cycles
L'attante réele sera entre 12 ms et 15 ms

b) **(0.75 points**) Expliquez brièvement le rôle de la minuterie dans la génération du Tick d'un système d'exploitation.

La minuterie est un compteur qui travaille avec la fréquence de l'horloge du système. Quand la valeur du compteur est à 0, un signal de sortie est généré : il s'agit d'une interruption qui est le signal de Tick pour le RTOS.

c) **(0.75 points)** La gestion de mémoire en MicroC est réalisée par une approche statique. Décrivez brièvement cette approche et expliquant clairement pourquoi il s'agit d'une approche statique (par opposition à l'approche dynamique).

MicroC utilise des partions de mémoire composées de blocs de même taille.
L'allocation de mémoire ce fait dans la partition correspondante à la taille mémoire nécessaire – contrairement à la location dynamique ou un premier espace libre est alloué.

Annexe

Contenu de OSUnMapTbl[]

```
INT8U const OSUnMapTbl[] = {
    0, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x00-0x0F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x10-0x1F
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x20-0x2F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x30-0x3F
    6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x40-0x4F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x50-0x5F
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x60-0x6F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x70-0x7F
    7, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x80-0x8F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x90-0x9F
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xA0-0xAF
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xB0-0xBF
    6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xC0-0xCF
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xD0-0xDF
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xE0-0xEF
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0   // 0xF0-0xFF
```

```
ptcb->OSTCBY     =  priority >> 3;
ptcb->OSTCBBitY  =  OSMapTbl[OSTCBY]
ptcb->OSTCBX     =  priority & 0x07;
ptcb->OSTCBBitX  =  OSMapTbl[OSTCBX];
```

Membres du TCB (ptcb est un pointeur sur le TCB traité)

OSMapTable

Index	Bit mask (Binary)
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

Opérations méttant une tâche dans la liste d'attente :

```
pevent->OSEventGrp           |= OSMapTbl[prio >> 3];
pevent->OSEventTbl[prio >> 3] |= OSMapTbl[prio & 0x07];
```

Code pour déterminer la tâche la plus prioritaire de la liste d'exécution

```
Y    =    OSUnMapTbl[OSRdyGrp] ;
X    =    OSUnMapTbl[OSRdyTbl[Y]] ;
prio =    [Y * 8] + X ;
```

Code to remove a task from the ready list / Code enlevant une tâche de la liste des tâches prêtes

```
if ((OSRdyTbl[ptcb->OSTCBY] &= ~ ptcb->OSTCBBitX) ==0)
{
        OSRdyGrp &= ~ ptcb->OSTCBBitY;
}
```

Code to make a task ready to run / Code qui rend une tâche prête à s'exécuter

```
OSRdyGrp              |=    ptcb->OSTCBBitY;
OSRdyTbl[ptcb->OSTCBY]|=    ptcb->OSTCBBitX;
```