

7 – ARCHITECTURES MATERIELLES – SYSTEMES D'EXPLOITATION

Exercices

1

Dans les années 1970, on pouvait placer 2000 transistors sur 10 mm².

Quelle surface aurait été nécessaire (en m²) pour placer 2 milliards de transistors ?

Même question pour la puce d'une clé USB de 2Go, en supposant que chaque bit soit mémorisé par un transistor.

2

Quels sont les spécificités et les avantages d'un système sur puce ?

3

Le premier smartphone, l'iPhone, est sorti en 2007. Comparer les caractéristiques du SoC équipant cet appareil avec le SoC de l'iPhone 12 (2020). On prendra comme critères la gravure, la fréquence, le nombre de cœurs du processeur, la taille des registres, la taille de la RAM, l'espace de stockage.

4 Consommation d'un processeur

1. Sous Windows, ouvrir le gestionnaire des tâches et un interpréteur Python. La consommation processeur du processus python.exe (en fait deux processus sont créés) est affichée à 0 %. Lancer le programme qui suit et observer alors la consommation processeur dans le gestionnaire des tâches :

```
for i in range(10000) :  
    print(i)
```

2. Commencer la lecture d'une vidéo. La consommation du processeur est-elle importante ? Dans l'affichage du gestionnaire des tâches, quelles sont les colonnes impactées ?

5 Lancement d'un processus avec Python

A l'aide de la fonction startfile du module os, afficher une page web.

6 Interblocage

On considère deux ressources R₁, R₂, et deux processus P₁, P₂, décrits ci-dessous :

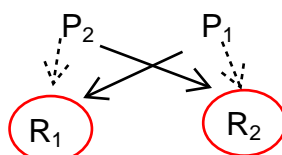
P₁ : demande R₁, demande R₂, libère R₁, libère R₂ ;

P₂ : demande R₂, demande R₁, libère R₂, libère R₁ .

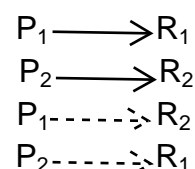
Si on exécute le processus P₁, puis P₂, il n'y a pas d'interblocage.

Décrire une exécution des deux processus conduisant à une situation d'interblocage, que l'on représentera par un schéma. On pourra représenter en traits pleins les demandes satisfaites, et en pointillés les demandes en attente.

P₂ puis P₁ en tourniquet



P₁ puis P₂ en tourniquet



7 .../

7 Interblocage

Considérons un petit système embarqué : un petit ordinateur relié à trois LED A , B et C . Une LED peut être éteinte ou allumée et on peut configurer sa couleur. On dispose de trois programmes qui affichent des signaux lumineux en faisant clignoter les LED. Chaque programme possède une LED primaire et une LED secondaire. Le programme P_1 affiche ses signaux sur A (primaire) et B (secondaire) en vert. Le programme P_2 affiche ses signaux sur B (primaire) et C (secondaire) en orange. Le programme P_3 affiche ses signaux sur C (primaire) et A (secondaire) en rouge. Comme les LED ne supportent pas d'être configurées dans deux couleurs en même temps, le système propose deux primitives `acquérirLED(nom)` et `rendreLED(nom)` qui permettent respectivement d'acquérir et de relâcher une LED. Si une LED est déjà acquise, alors `acquérirLED()` bloque.

On suppose que chacun des trois programmes P_1 , P_2 et P_3 effectue les actions suivantes en boucle :

1. acquérir sa LED primaire
2. acquérir sa LED secondaire
3. configurer les couleurs
4. émettre des signaux
5. rendre la LED secondaire
6. rendre la LED primaire
7. recommencer en 1

Montrer qu'il existe un entrelacement des exécutions qui place P_1 , P_2 et P_3 en interblocage.

En tourniquet

```
P1 --> A
P2 --> B
P3 --> C
P1 -/-> B --> interblocage
P2 -/-> C
P3 -/-> A
```

https://www.electronique-et-informatique.fr/Electronique-et-Informatique/Digit/Proc_ordi/Pratiques_Ordinateurs/Fonction_du_Z80.php

<https://www.electronique-et-informatique.fr/Electronique-et-Informatique/index.php>

