

Chapitre 19 - Architecture ordinateur- OS

Les ordinateurs sont gérés par un système d'exploitation. L'O.S. (**O**perating **S**ystem) gère le chargement des programmes depuis la mémoire de masse et le lancement de leur exécution en créant des processus. Il gère aussi l'ensemble des ressources matérielles de l'ordinateur.

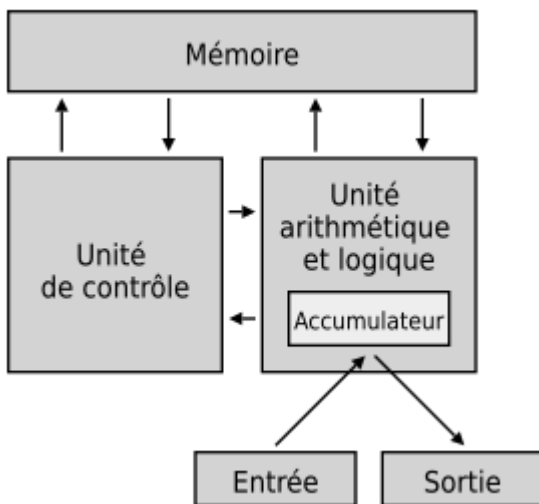
On débute ce chapitre par une présentation de l'architecture matérielle des ordinateurs. On s'intéresse ensuite à l'OS et aux différentes méthodes de gestion des processus. On termine par une présentation des puces Soc.

1- ARCHITECTURE D'UN ORDINATEUR :

L'architecture des ordinateurs actuels est similaire à celle établie en 1945 par John Von Neumann, utilisée pour réaliser un des tout premier ordinateur électronique (photo ci-contre)

L'**architecture de von Neumann** décompose l'ordinateur en 4 parties distinctes :

- l'unité arithmétique et logique (UAL ou ALU en anglais) ou unité de traitement : son rôle est d'effectuer les opérations de base ;



- l'unité de contrôle, chargée du « séquençage » des opérations ;

- la mémoire qui contient à la fois les données et le programme indiquant à l'unité de contrôle quels calculs sont à faire sur ces données. La mémoire se divise entre mémoire volatile (programmes et données en cours de fonctionnement) et mémoire permanente (programmes et données de base de la machine) ;

- les dispositifs d'entrée-sortie, qui permettent de communiquer avec le monde extérieur.



2- PRINCIPAUX COMPOSANTS :

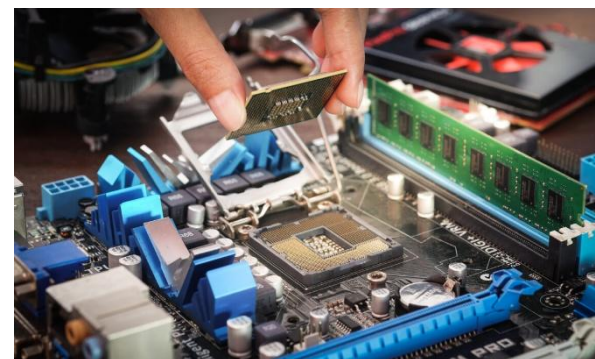
a. Le processeur :

Le **processeur** (*Central Processing Unit, CPU*) est le composant essentiel qui exécute les instructions des programmes informatiques.

Remarque : il est le plus souvent amovible, placé sur un support appelé **socket**, et équipé d'un **radiateur** et d'un **ventilateur** (c'est le composant de la carte mère le plus gourmand en énergie).

Il est schématiquement constitué de 3 parties :

- o l'**unité arithmétique et logique** (ALU) est chargée de l'exécution de tous les calculs : opération arithmétiques (sur les nombres entiers ou flottants) et opérations logiques (sur les bits) ;



- les **registres** permettent de mémoriser de l'information (donnée ou instruction) au sein même du CPU, en très petite quantité ;
- l'**unité de contrôle** permet d'exécuter les instructions (les programmes) elle joue le rôle de « chef d'orchestre » .

b. La mémoire :

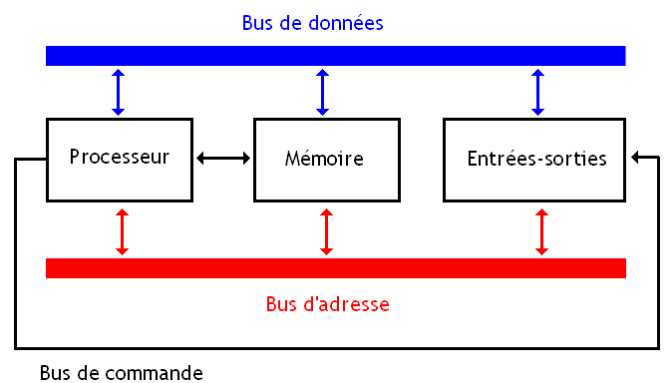
La **mémoire** permet de stocker des données et des programmes. On distingue :

- La mémoire permanente (technologie SSD ou disque dur)
- La mémoire vive (RAM) , mémoire volatile, qui est l'espace de stockage principal du processeur,
- La mémoire cache, mémoire plus rapide que la RAM qui sert à conserver un court instant des informations consultées fréquemment,
- Les registres de processeur, mémoires intégrées au processeur, de petites capacités, mais extrêmement rapides.

c. Les Bus:

Pour que les données circulent entre les différentes parties d'un ordinateur (mémoire, CPU et les entrées/sorties), il existe des systèmes de communication appelés **bus**. Il en existe de 3 grands types :

- Le **bus d'adresse** permet de faire circuler des adresses *par exemple l'adresse d'une donnée à aller chercher en mémoire ;*
- Le **bus de données** permet de faire circuler des données ;
- Le **bus de contrôle** permet de spécifier le type d'action



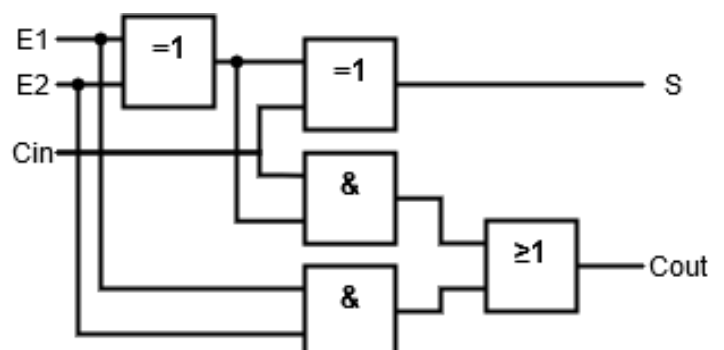
d. Circuits logiques pour une addition :

Dans les processeurs, les opérations booléennes sont réalisées par des transistors formant ce que l'on appelle des

L'addition de 2 bits peut par exemple être réalisé par une porte logique XOR :

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Pour additionner des nombres codés sur plusieurs *bits* (comme les entiers naturels par exemple), il faut réaliser plusieurs additions de *bits*, mais en tenant compte de la retenue :

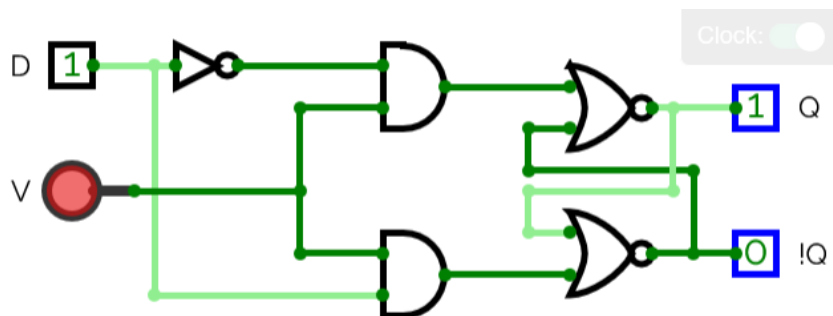


e. Circuits logiques pour une mémoire:

Dans la RAM, il existe plusieurs circuits logiques, appelés **bascules**, permettant de mémoriser des *bits* :

Exemple : l'entrée D correspond au *bit* à mémoriser. Pour effectuer la mémorisation, il faut activer le bit de validation V. La sortie Q prend alors la valeur de D à l'instant de la validation, lorsque V passe de 0 à 1 (on parle de **front montant** de V).

Tant que V reste à 0, Q garde la valeur mémorisée, quelle que soit la valeur de D.



f. Instructions machines – Assembleur :

Le CPU ne gère que des grandeurs booléennes : les instructions exécutées au niveau du CPU sont donc codées en binaire. L'ensemble des instructions exécutables directement par le processeur constitue ce que l'on appelle le **langage machine**.

Le microprocesseur étant incapable d'interpréter la phrase « *additionne le nombre 125 et la valeur située dans le registre R2, range le résultat dans le registre R1* », il faut coder cette instruction sous forme binaire :

« *additionne le nombre 125 et la valeur située dans le registre R2, range le résultat dans le registre R1* » en binaire donnera : « 11100010100000100001000001111101 »

Afin de faciliter la lecture et l'écriture d'instructions machine par les informaticiens, on remplace les codes binaires par des **symboles mnémoniques**, en utilisant la syntaxe du langage appelé **assembleur**.

« **additionne le nombre 125 et la valeur située dans le registre R2, range le résultat dans le registre R1** »

donnera en assembleur « **ADD R1,R2,#125** »

ce qui donnera en binaire « 11100010 10000010 00010000 01111101 »

3- SYSTEME D'EXPLOITATION :

Un **système d'exploitation** (ou **OS** pour **Operating System**) est un ensemble de programmes permettant de gérer l'ordinateur : gestion des matériels et des processus à exécuter pour les différents programmes ou applications lancées.

Dans le secteur informatique, les systèmes d'exploitation les plus répandus sont **Windows** (pour les PC), **Mac OS** (pour les ordinateurs d'Apple), **Linux** (pour les PC et les serveurs) et **Unix** (pour les serveurs). Sur smartphone, on trouve **Android** qui utilise un noyau Linux (open Source) et **IOS** d'Apple.

En 2025, Android est le système d'exploitation le plus utilisé au monde (si l'on prend en compte l'utilisation Web). Il détient 43 % du marché mondial, suivi de Windows (30 %), d'Apple iOS (17 %), de macOS (6 %), puis de Linux (desktop) (0,98 %), qui utilise également le noyau Linux⁴. Ces chiffres ne tiennent pas compte des consoles de jeux. Pour les smartphones et autres appareils de poche, Android est en tête avec 72 % de parts de marché, et iOS d'Apple en détient 28 %. Pour les ordinateurs de bureau et les ordinateurs portables, Windows est le plus utilisé (76 %), suivi de macOS d'Apple (17 %) et des systèmes d'exploitation basés sur Linux (4 %) (c'est-à-dire « Linux desktop », 3,22 %, plus ChromeOS de Google, 1,04 %).

4- LES PROCESSUS :

a. C'EST QUOI UN PROCESSUS ? :

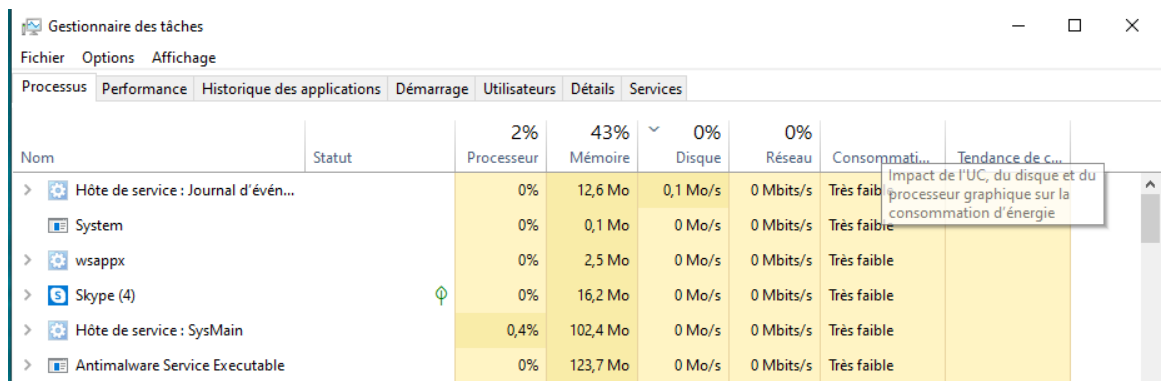
Point Cours :

Un processus est une instance (tâche) de programme en cours d'exécution sur un ordinateur. Il est caractérisé par :

- un ensemble d'instructions à exécuter - souvent stockées dans un fichier sur lequel on clique pour lancer un programme (par exemple firefox.exe)
- un espace mémoire dédié à ce processus pour lui permettre de travailler sur des données et des ressources qui lui sont propres : si deux instances de firefox sont lancées, chacune travaillera indépendamment l'une de l'autre.
- des ressources matérielles : processeur, entrées-sorties (accès à internet en utilisant la connexion Wifi).

Sur chaque OS, des utilitaires permettent de lister les processus en cours d'exécution :

Sur **Windows**, on ouvre le gestionnaire de tâches :



Nom	Statut	Processeur	Mémoire	Disque	Réseau	Consommation d'énergie	Tendance de consommation d'énergie
Hôte de service : Journal d'évén...		0%	12,6 Mo	0,1 Mo/s	0 Mbits/s	Très faible	Impact de l'UC, du disque et du processeur graphique sur la consommation d'énergie
System		0%	0,1 Mo	0 Mo/s	0 Mbits/s	Très faible	
wsappx		0%	2,5 Mo	0 Mo/s	0 Mbits/s	Très faible	
Skype (4)		0%	16,2 Mo	0 Mo/s	0 Mbits/s	Très faible	
Hôte de service : SysMain		0,4%	102,4 Mo	0 Mo/s	0 Mbits/s	Très faible	
Antimalware Service Executable		0%	123,7 Mo	0 Mo/s	0 Mbits/s	Très faible	

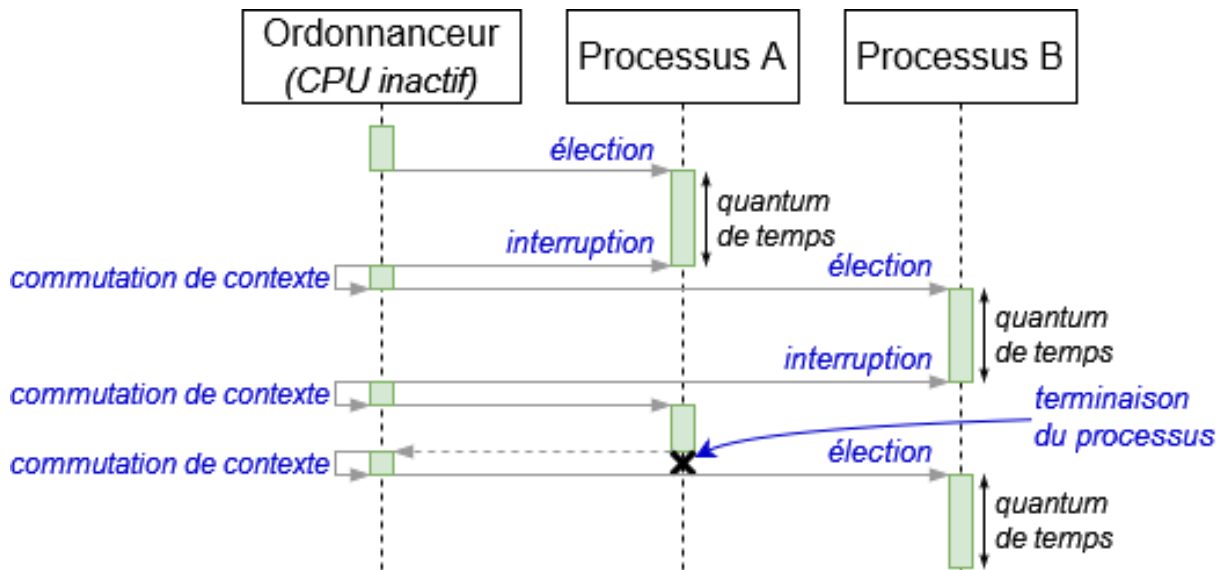
Sur un terminal de commande **Linux**, on peut exécuter la commande **ps** :

```
[root@tecmint ~]# ps -ef
UID          PID    PPID  C  STIME TTY          TIME CMD
root         1      0  0  13:04 ?        00:00:01 /usr/lib/systemd/systemd --switc
root         2      0  0  13:04 ?        00:00:00 [kthreadd]
root         4      2  0  13:04 ?        00:00:00 [kworker/0:0H]
root         5      2  0  13:04 ?        00:00:00 [kworker/u2:0]
root         6      2  0  13:04 ?        00:00:00 [mm_percpu_wq]
root         7      2  0  13:04 ?        00:00:00 [ksoftirqd/0]
root         8      2  0  13:04 ?        00:00:00 [rcu_sched]
root         9      2  0  13:04 ?        00:00:00 [rcu_bh]
root        10      2  0  13:04 ?        00:00:00 [rcuos/0]
root        11      2  0  13:04 ?        00:00:00 [rcuob/0]
```

- **PID :**
- **PPID :**

b. GESTION DE PLUSIEURS PROCESSUS :

Les OS sont **multitâches** et peuvent gérer plusieurs processus lancés en même temps. Sachant que le processeur (ou un cœur de processeur) ne peut exécuter qu'une seule instruction à la fois, l'OS organise la charge de travail du processeur, en lui demandant de basculer constamment d'un processus à l'autre.

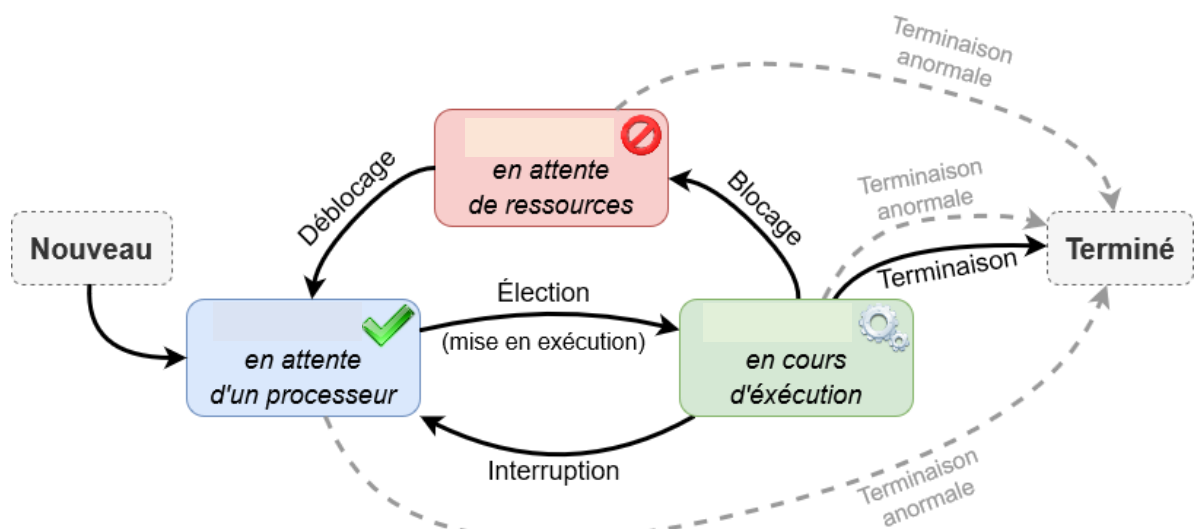


c. ETAT D'UN PROCESSUS :

Lorsqu'un programme est lancé, les instructions machine qui le composent sont chargées en mémoire de travail (RAM) : le processus associé est alors créé.

Pendant son existence au sein d'une machine, un processus peut avoir différents états :

- **nouveau** : le processus est en cours de création, l'exécutable est en mémoire et le PCB initialisé
- **prêt (ready ou runnable)** ou en **attente (waiting)** : le processus attend d'être affecté à un processeur
- **élu (running)** : les instructions du processus sont en cours d'exécution (il utilise le CPU) seul un processus peut être en exécution sur processeur à un instant donné.
- **bloqué (blocked)** ou **endormi (sleeping)** : le processus est interrompu en attente qu'un événement se produise
- **terminé (terminated)** : le processus est terminé (soit normalement, soit suite à une anomalie). il doit être déchargé de la mémoire par l'OS, et les ressources qu'il utilisait libérées.



d. ORDONNANCEMENT DES PROCESSUS :

La gestion des états des processus est confiée à un programme de l'OS, appelé (**scheduler**). L'utilisation du processeur est optimisée : un processus en attente d'une ressource (donnée en mémoire, entrée/sortie, ...) est mis dans l'*état bloqué* pour permettre à un autre processus en attente d'un processeur (*état prêt*), de passer dans l'*état élu*.

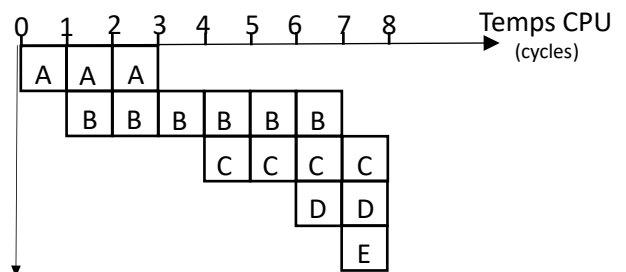
Lorsque plusieurs processus sont en attente (*état prêt*), l'ordonnanceur doit choisir le processus à élire : il les classe dans une *file d'attente*. L'ordonnanceur de l'OS sélectionne un processus dans la file d'attente et le laisse s'exécuter pendant un délai maximum déterminé appelé le **quantum de temps**. À la fin de ce temps, ce processus est remis à l'*état prêt* et un autre est élu.

L'ordonnanceur peut envisager diverses politiques d'ordonnancement :

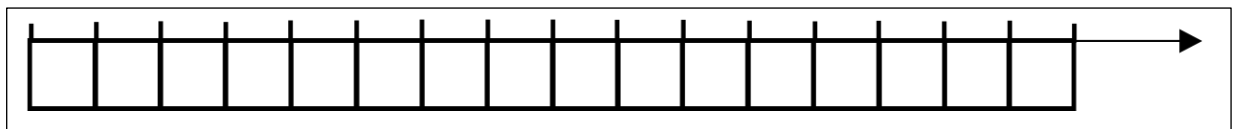
- **Premier arrivé, premier servi** : simple, mais peu adapté
- **Plus court d'abord** : efficace, mais difficile de classer à l'avance la durée d'exécution d'un processus
- **Priorité** : chaque processus se voit attribuer un niveau de priorité
- **Tourniquet** : chaque processus se voit attribuer une durée d'exécution (appelé **quantum de temps**). Quand un processus a atteint cette durée, il cède la place à un autre. Dans le détail, on a le fonctionnement suivant à chaque cycle :
 - si un nouveau processus est créé, il est mis dans la file d'attente en début de cycle,
 - ensuite, on défile un processus de la file d'attente et on l'exécute durant un cycle,
 - si le processus exécuté n'est pas terminé, on le replace dans la file d'attente.
- ... politiques hybrides ...

Appliqué à l'exemple donné ci-dessous, ces différentes politiques donnent des organisations différentes. Dans cet exemple, on suppose que le temps de commutation est nul.

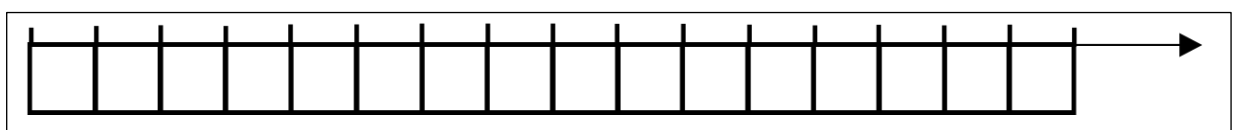
Processus	Temps d'exécution	Temps d'arrivage
A	3	0
B	6	1
C	4	4
D	2	6
E	1	7



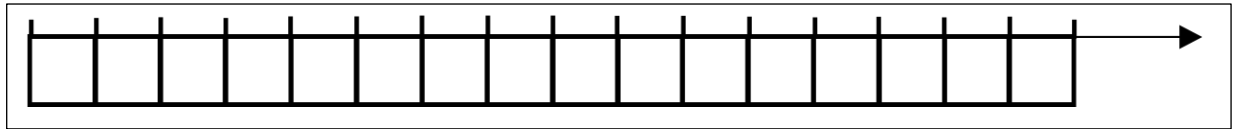
⇒ Ordonner ces processus suivant l'algorithme "**Premier arrivé premier servi**" : **First-Come First-Served (FCFS)**, en supposant qu'il n'y pas de blocage.



⇒ Ordonner ces processus suivant l'algorithme "**Le plus court d'abord**" : **Short Job First (SJF)**

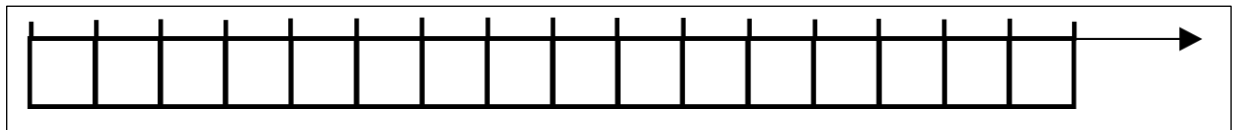


⇒ Ordonnancer ces processus avec **Algorithme du Tourniquet** en prenant un quantum de 1 unités de temps :



[0 – 1] → _____ →	[5 – 6] → _____ →	[10 – 11] → _____ →
[1 – 2] → _____ →	[6 – 7] → _____ →	[11 – 12] → _____ →
[2 – 3] → _____ →	[7 – 8] → _____ →	[12 – 13] → _____ →
[3 – 4] → _____ →	[8 – 9] → _____ →	[13 – 14] → _____ →
[4 – 5] → _____ →	[9 – 10] → _____ →	[14 – 15] → _____ →

⇒ Ordonnancer ces processus avec **Algorithme du tourniquet** en prenant un quantum de 2 unités de temps :



[0 – 1] → _____ →	[5 – 6] → _____ →	[10 – 11] → _____ →
[1 – 2] → _____ →	[6 – 7] → _____ →	[11 – 12] → _____ →
[2 – 3] → _____ →	[7 – 8] → _____ →	[12 – 13] → _____ →
[3 – 4] → _____ →	[8 – 9] → _____ →	[13 – 14] → _____ →
[4 – 5] → _____ →	[9 – 10] → _____ →	[14 – 15] → _____ →

Ordonnancement préemptif ou non préemptif :

L'ordonnancement peut être préemptif ou non. Il est dit non préemptif si, **une fois que le processeur a été alloué à un processus, il le gardera jusqu'à sa terminaison (temporaire ou définitive)**. Il est préemptif dans le cas contraire.

e. INTERBLOCAGE DES PROCESSUS :

Interblocage :

Un **interblocage** (ou étreinte fatale, *deadlock* en anglais) est un phénomène qui se produit lorsque des processus concurrents s'attendent mutuellement. Les processus bloqués dans cet état le sont définitivement, il s'agit donc d'une situation catastrophique.

Exemple :

Supposons un système possédant les ressources 1 et 2, ayant les processus A et B en exécution. Supposons aussi que les ressources ne sont pas partageables : un seul processus à la fois peut utiliser une ressource.

Une façon de provoquer un interblocage est la suivante :

- Le processus A utilise la ressource 1;
- Le processus B utilise la ressource 2;
- Le processus A demande la ressource 2, et devient à l'état bloqué.
- Le processus B demande la ressource 1, et devient à l'état bloqué.

Modélisation des situations d'interblocage :

On modélise les interblocages à l'aide de graphes orientés conçus de la façon suivante :

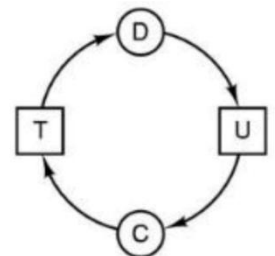
- Les **processus** sont représentés par des **cercles**.
- Les **ressources** sont représentées par des **carrés**.
- Une flèche qui va d'un carré à un cercle indique que la ressource est déjà attribuée au processus (figure a).
- Une flèche d'un cercle vers un carré indique que le processus est bloqué en attente de cette ressource (figure b).
- Les **interblocages** sont représentés dans ces graphiques par la présence d'un circuit dans le graphe orienté (figure c).



(a)



(b)



(c)

Exemple détaillé d'une situation

d'interblocage : P1 et P2 sont 2 processus qui doivent utiliser les ressources R1 et R2 dans l'ordre

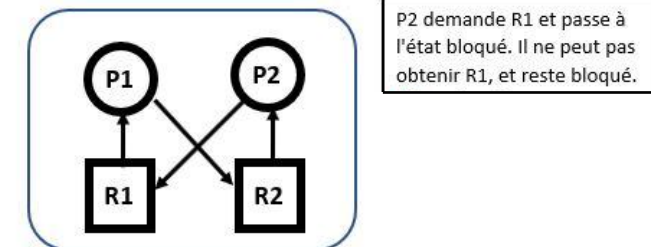
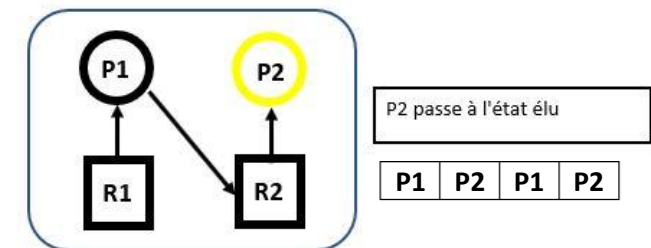
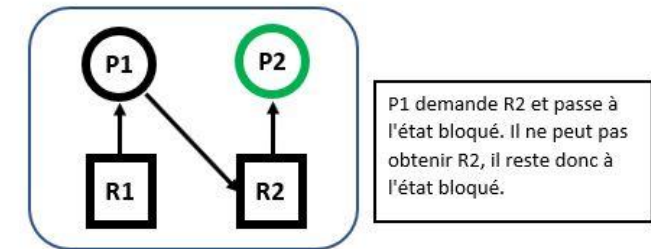
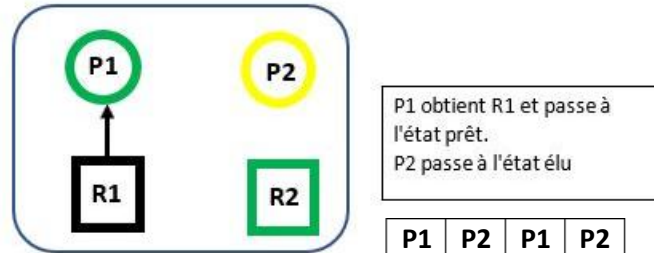
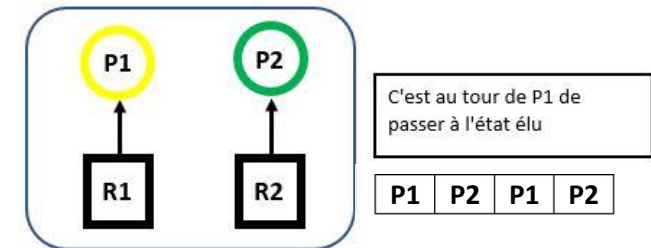
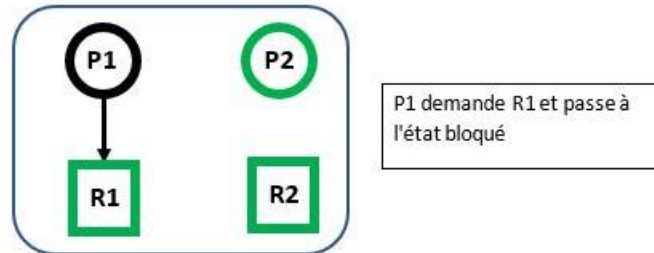
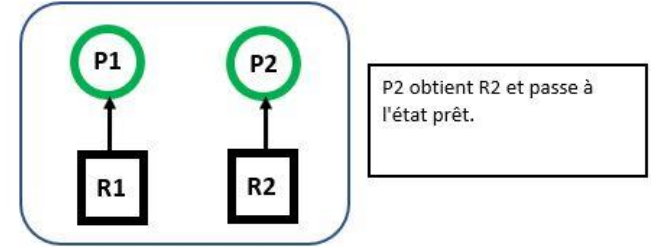
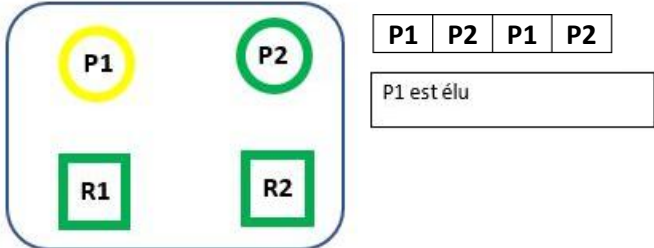
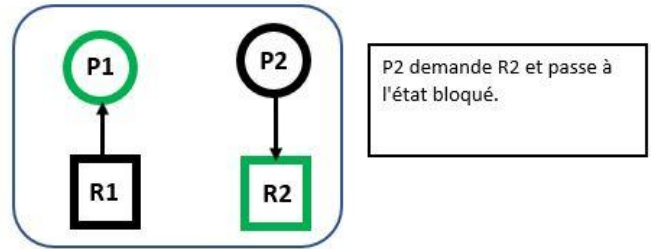
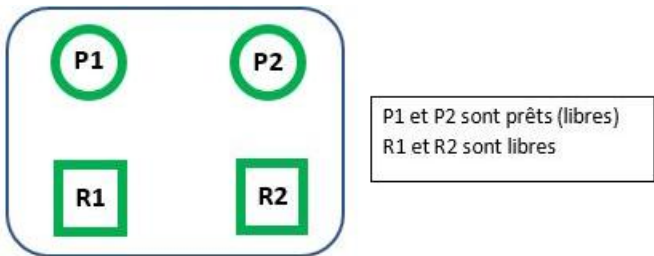
On suppose que les ressources R1 et R2 sont à **usage exclusif** : elles sont soit disponibles, soit attribuées à un unique processus.

<i>Processus P1</i>	<i>Processus P2</i>
Accéder à la ressource R1	Accéder à la ressource R2
Accéder à la ressource R2	Accéder à la ressource R1

On suppose que l'ordonnancement suit un algorithme de **Tourniquet**, qui alterne les instructions des processus, ici sur 4 quantums

P1	P2	P1	P2
----	----	----	----

L'exécution détaillée des 4 instructions donnera :



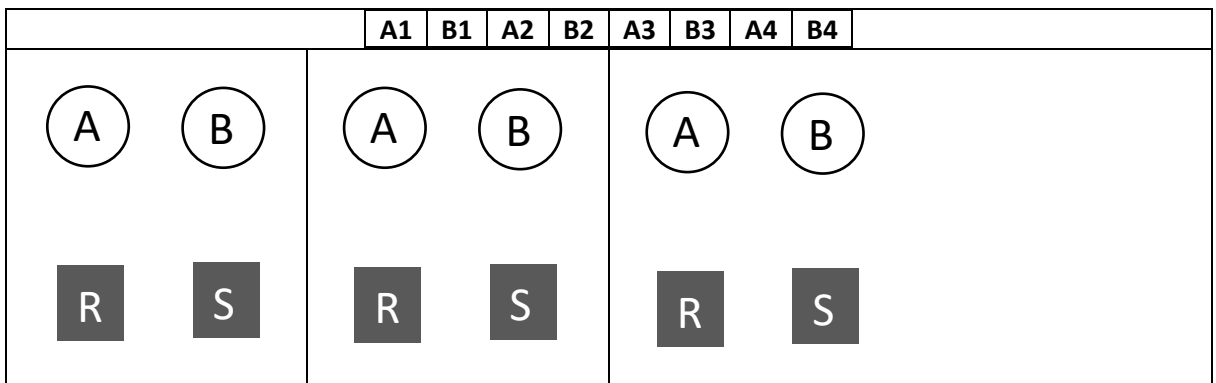
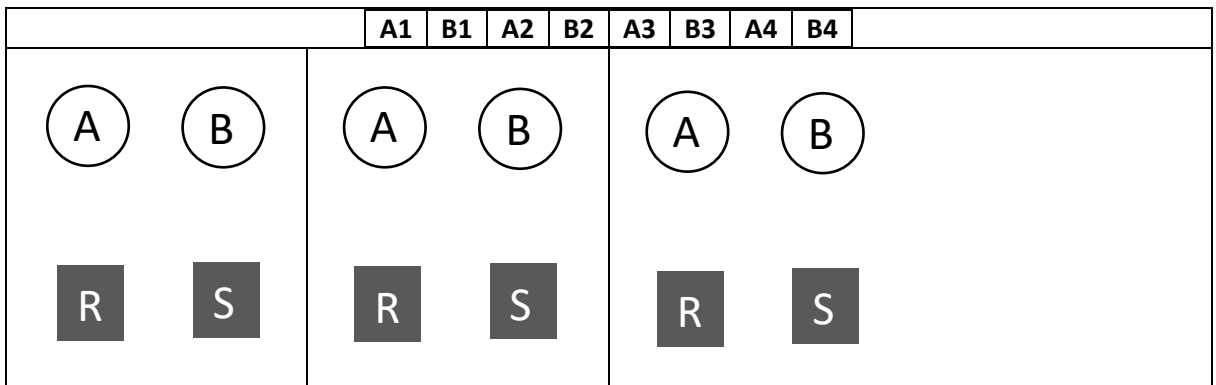
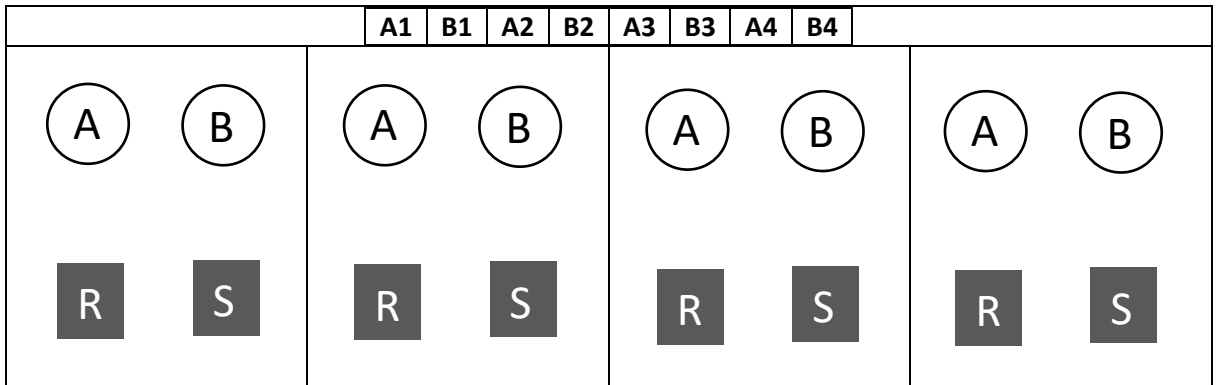
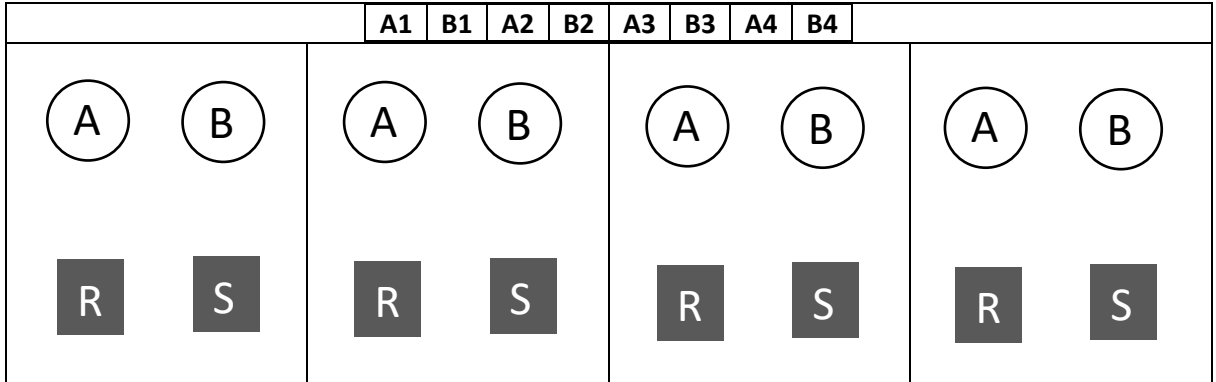
Sur cette dernière étape, la situation totalement bloquée et l'on se retrouve dans une situation d'**interblocage**.

Exercice : On considère 2 processus A et B, et deux ressources R et S à usage exclusif. L'action des processus A et B est décrite ci-contre :

A1	B1	A2	B2	A3	B3	A4	B4
----	----	----	----	----	----	----	----

⇒ Montrer que cet ordonnancement conduit à une situation d'interblocage :

Processus A	Processus B
étape A1 : demande R	étape B1 : demande S
étape A2 : demande S	étape B2 : demande R
étape A3 : libère S	étape B3 : libère R
étape A4 : libère R	étape B4 : libère S



f. APPARITION ET PREVENTION DES INTERBLOCAGES:

Les situations d'interblocage ont été théorisées par l'informaticien Edward Coffman (1934-) qui a énoncé quatre conditions - appelées conditions de Coffman - menant à l'interblocage :

Conditions qui mènent à un interblocage :

- **Exclusion mutuelle** : Les ressources ne sont pas partageables, un seul processus à la fois peut utiliser la ressource.
- **Rétention des ressources** : un processus détient au moins une ressource et requiert une autre ressource détenue par un autre processus. C'est à dire que les processus qui détiennent des ressources peuvent en demander d'autres.
- **Non préemption** : Seul le processus détenteur d'une ressource peut la libérer. On ne peut pas forcer un processus à rendre une ressource.
- **Attente circulaire** : Chaque processus attend une ressource détenue par un autre processus. P1 attend une ressource détenue par P2 qui à son tour attend une ressource détenue par P3 etc... qui attend une ressource détenue par P1 ce qui clos la boucle.

Il existe heureusement des stratégies pour éviter ces situations. Par exemple :

- la prévention : on oblige le processus à déclarer à l'avance la liste de toutes les ressources auxquelles il va accéder.
- l'évitement : on fait en sorte qu'à chaque étape il reste une possibilité d'attribution de ressources qui évite le deadlock.
- la détection/résolution : on laisse la situation arriver jusqu'au deadlock, puis un algorithme de résolution détermine quelle ressource libérer pour mettre fin à l'interblocage.

5- LES SOCS :

a. L'INTEGRATION DE COMPOSANTS DIFFERENTS AU SEIN D'UNE MEME PUCE :

Le principe d'un **système sur puce** ou **System On a Chip** (SoC) est d'intégrer au sein d'une puce unique, un ensemble de composants habituellement physiquement dissociés dans un ordinateur classique (ordinateur de bureau ou ordinateur portable).

On peut retrouver ainsi au sein d'une **même puce** :

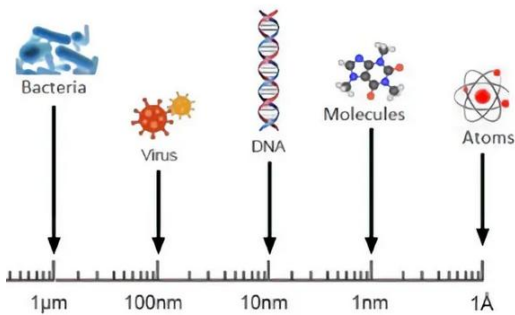
- le microprocesseur (CPU : **C**entral **P**rocessing **U**nit)
- la carte graphique (GPU : **G**raphics **P**rocessing **U**nit)
- la mémoire RAM
- une unité dédiée aux calculs d'intelligence artificiel (NPU)

Avantages : moindre consommation électrique, pas besoin de refroidissement, moindre encombrement, moindre coût.

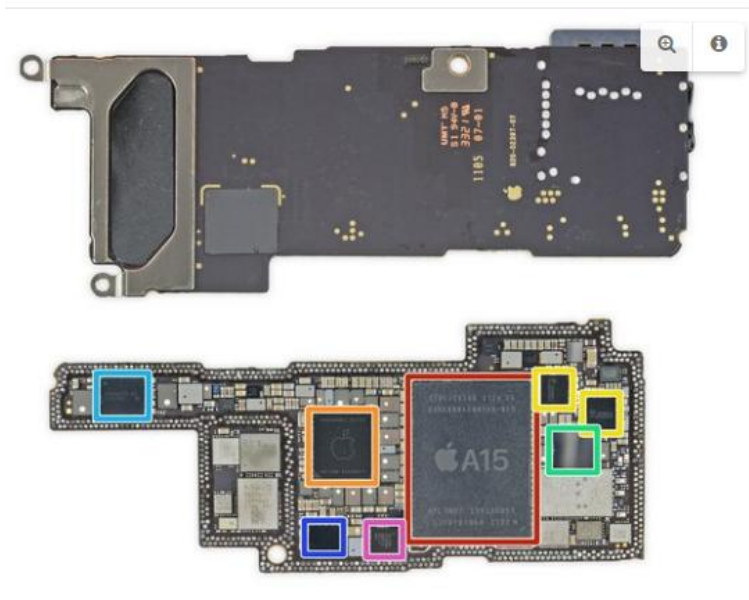
Inconvénients : moins de modularité, moins de réparabilité.

Exemple : puce de l'iPhone

« Apple a pris une avance considérable en matière de SoC depuis qu'elle a décidé de concevoir elle-même ses propres puces pour ses iPhone 15, iPad et Mac. Elle bénéficie ainsi d'une meilleure intégration entre le matériel et le logiciel, ce qui lui permet d'optimiser les performances et l'autonomie de ses appareils. Elle profite également du savoir-faire du fabricant de circuits intégrés taiwanais TSMC, qui fabrique ses puces selon des procédés toujours plus avancés, et dont l'accord entre les deux parties permet d'économiser plusieurs milliards à Apple. »



	A18 Pro iPhone 16 Pro	A18 iPhone 16	A17 Pro iPhone 15 Pro
CPU	6 cœurs (2P+4E)	6 cœurs (2P+4E)	6 cœurs (2P+4E)
GPU	6 cœurs	5 cœurs	6 cœurs
Mémoire	8 Go LPDDR5	8 Go LPDDR5	8 Go LPDDR5
Stockage	128 à 1024 Go	128 à 1024 Go	128 à 1024 Go
NPU	16 cœurs 35 TOPS	16 cœurs 35 TOPS	16 cœurs 35 TOPS
USB (ou Lightning)	USB 3 (10 Gb/s)	USB 2 (480 Mb/s)	USB 3 (10 Gb/s)
Gravure	3 nm 2e génération	3 nm 2e génération	3 nm
Nombre transistors	NC		19 milliards



● Carte mère, face 1 :

- Apple APL1W07 A15 Bionic SoC superposé avec ce qui sont très probablement 6 Go de SK Hynix LPDDR4X SDRAM
- CI de gestion de la puissance Apple APL1098
- CI de gestion de la puissance Apple 338S00762-A1
- CI de gestion de la puissance STMicroelectronics STB601A05
- CI de gestion de la puissance Apple 338S00770-B0
- CI de gestion de la puissance de l'écran Texas Instruments TPS65657B0
- Multiplexeur de port d'écran NXP Semiconductor CBTL1616A0

b. UN PRINCIPE GENERAL :

Certains processeurs sont optimisés pour certains types de calcul. C'est le cas par exemple d'une carte graphique, qui excelle dans le calcul de rendus de polygones. On s'est aperçu que cette aptitude à faire des calculs « bêtes et répétitifs » était parfaite pour faire les calculs mathématiques nécessaires

au minage des cryptomonnaies. Les cartes graphiques ont donc été détournées de leur usage originel. De même, les calculs sur les réseaux de neurones (essentiels en IA) nécessitent une grande rapidité dans les multiplications de matrices. Pour cette raison, Apple a intégré directement dans son SoC A15 une puce spécialisée pour ces calculs.

c. CONCLUSION :

L'orientation actuelle de l'électronique est donc à la fois :

- une intégration toujours plus grande dans des SoC multi-tâches.
- des puces toujours plus spécifiques qui excellent dans un domaine particulier.

d. EXERCICE : QUESTION BAC 2021

Un constructeur automobile intègre à ses véhicules des systèmes embarqués, comme par exemple un système de guidage par satellites (GPS), un système de freinage antiblocage (ABS)...

Ces dispositifs utilisent des systèmes sur puces (SoC : System on a Chip).

Question : Citer deux avantages à utiliser ces systèmes sur puces plutôt qu'une architecture classique d'ordinateur.