CH4 Architecture matérielle

1 Les SoC

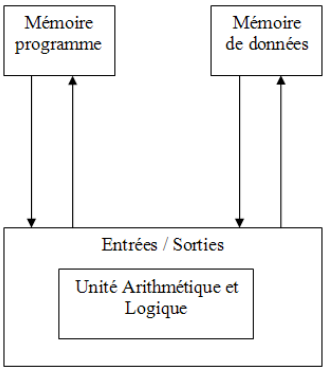
(a) Architecture de Harvard

Au sein d'**un ordinateur « classique »** tel qu'un PC de bureau, les choses sont assez simples :

- **Le processeur (CPU)** se charge de réaliser les calculs les plus répandus, ceux qui permettent par exemple de faire tourner le système d'exploitation ou un navigateur web.
- **La carte graphique (ou GPU)** se charge d'afficher une image, qu'elle soit en 2D ou bien en 3D comme dans les jeux.
- **La carte-mère** relie entre eux tous les composants, le CPU, le GPU, mais également la RAM et d'autres cartes additionnelles et petites puces

A retenir :
Les ordinateurs actuels utilisent l'**architecture de Harvard**.
C'est à dire que les données et les programmes occupent des parties différentes de la mémoire et accèdent au CPU par des bus disctincts.

Ainsi, le CPU peut **traiter en parallèle des données et des instructions**, ce qui permet de gagner du temps.

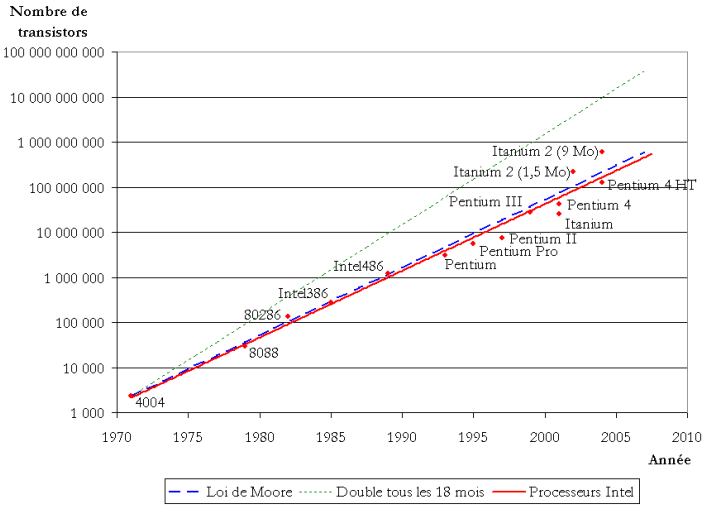


Remarque :
Voir https://fr.wikipedia.org/wiki/Architecture_de_von_Neumann et https://fr.wikipedia.org/wiki/Architecture_de_type_Harvard pour comparer les deux.

En 1975, Gordon E. Moore, ingénieur chez Intel, énonça :

« Dans les microprocesseurs, le nombre de transistors sur une puce va doubler tous les 2 ans. »

Cette loi (dite *loi de Moore*) s'est globalement vérifiée et a permis de gros progrès de miniaturisation.

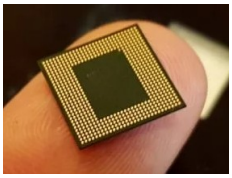


Voir https://www.futura-sciences.com/tech/definitions/informatique-loi-moore-2447/)

(b) Un système tout en un

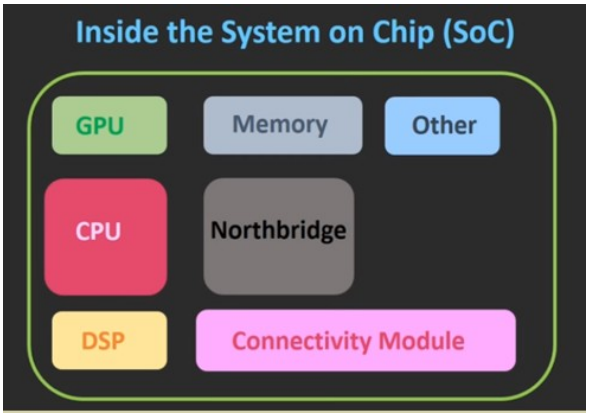
En poussant encore cette loi de Moore, depuis le début de l'ère des smartphones, on assiste à l'émergence de systèmes tout-en-un.
Ainsi, presque tout le contenu d'un ordinateur se retrouve finalement dans une seule puce sur le smartphone : **le System on a Chip (SoC),** ou système sur une puce en français. C'est un circuit intégré essentiel au fonctionnement des objets connectés et des smartphones.

Le système sur une puce comprend tous les éléments essentiels d'un ordinateur comprimé dans une forme réduite. Son **faible encombrement**, son caractère complet et sa f**aible consommation d'énergie** en font un circuit intégré idéal pour les applications mobiles, notamment l'IoT.



A retenir :
Un SoC comprend à la fois:

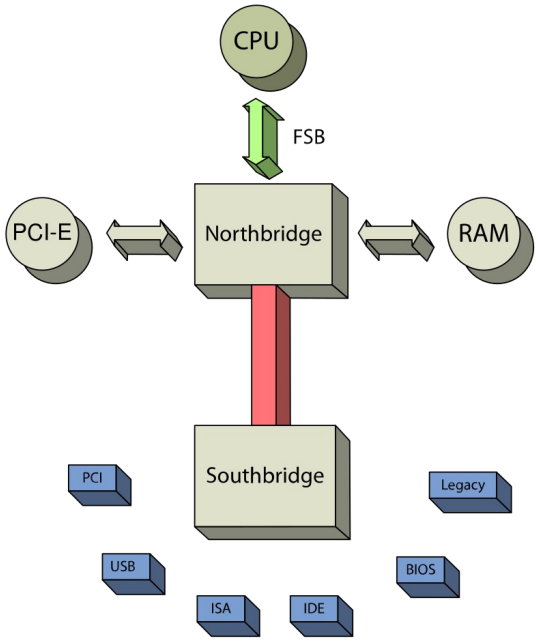
- le processeur central à un ou plusieurs cœurs de calcul **CPU**
- un processeur graphique **GPU**
- un processeur de signal numérique **DSP**
- la mémoire vive et la mémoire statique (Rom, Flash, EPROM) **Memory**
- les puces de communications (Bluetooth, WiFi, 2G/3G/4G, LoRa…) et les capteurs nécessaires au fonctionnement d'un smartphone ou d'un objet connecté **Connectivity Module**
- ... Autre



https://www.youtube.com/watch?v=FUhCrWoNA2c

Remarque :
Dans le schéma (ci-dessus), on voit apparaître « ***Northbridge*** » qui est la partie de la carte mère connectée directement au CPU.

Les deux parties ***Northbridge*** et ***Southbridge*** sont intimement liées et forment le ***chipset*** qui lie les éléments entre eux. Dans les SoC, on retrouve donc cette architecture.



Exemples :
→ Voici les noms de SoC utilisés par quelques produits à succès :

- Apple iPhone 11 : SoC A13 Bionic
- Nintendo Switch : SoC Nvidia Tegra
- Apple MacBook : SoC M1

→ SoC Exynos 990 qui équipe les **Galaxy_S20**

On y trouve:

- **Le processeur** (CPU)
- **Le carte graphique** (GPU)
- **La puce neuronale** ou Neutral Processing Unit (NPU), est une puce en charge de l'intelligence artificielle du smartphone.
- **Le modem** qui gère non seulement le Wifi, Bluetooth, le NFC ou bien encore les technologies mobiles (4G, 5G ou 3G).
- **Le processeur de signal numérique** ou Digital Signal Processor (DSP) est en charge de traiter les signaux numériques. Permet le filtrage, la compression ou encore l'extraction de différents signaux (musique, vidéo, ...).
- **Le processeur d'image** ou Image Signal Processor (ISP) est une puce prenant en charge la création d'images numériques. C'est grâce à elle que le smartphone va traiter la prise de photo. Elle est intégrée au GPU.
- **Le processeur sécurité** ou Secure Processing Unit (SPU) est le bouclier du smartphone. Son alimentation électrique est indépendante afin de ne pas pouvoir être éteint en cas d'attaque sur celui-ci. Le SPU va stocker vos données biométriques, bancaires, SIM, etc.

(c) Les différents types de SoC

Les utilisations variées de ce type de circuit intégré demande des architectures sensiblement différente. On distingue **3 grandes familles de SoC** :

- Le système sur une puce construit autour d'un **microcontrôleur**, la forme la plus simple d'un SoC qui a donné naissance aux cartes Arduino.
- Le système sur une puce construit autour d'un **microprocesseur**. Il s'agit du SoC, le plus répandu parce qu'utilisé par tous les fabricants de smartphones. Doté d'un Bus externe, il permet de connecter de nombreux capteurs.
- Le système sur une **puce dédié à une tâche spécifique**. Cette dernière famille comprend notamment les puces reprogrammables FPGA.

L'Internet des objets peut faire intervenir les trois familles de système sur une puce, suivant la complexité de l'objet, du capteur ou du système embarqué connecté à concevoir. Cependant, le Soc basé sur un microprocesseur prend généralement place dans la plupart des objets connectés.

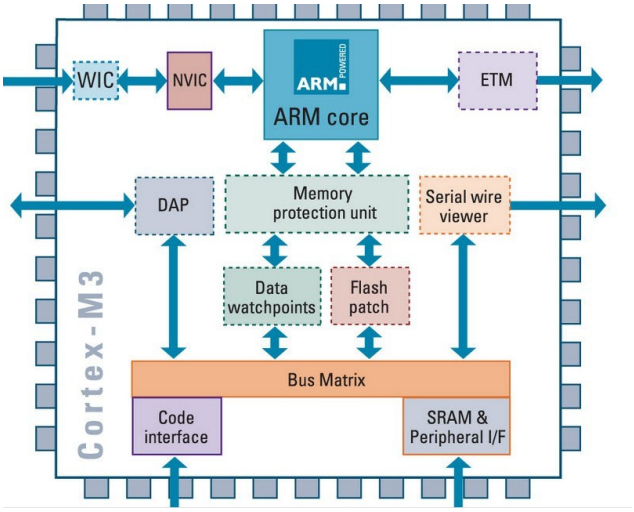
Suivant les fabricants et les besoins, il existe **plusieurs architectures**. Deux d'entre elles prennent place dans la grande majorité des produits électroniques conçus ces vingt dernières années.

- **L'architecture ARM :**

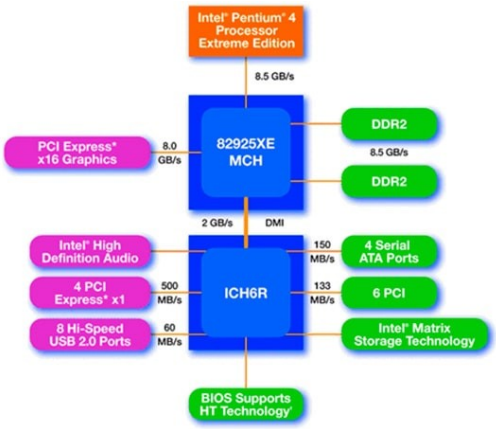
Conçu par la société du même nom, l'architecture ARM a été développé en interne à la fin des années 1980. C'est en 1987 qu'elle est la première fois utilisée dans la gamme d'ordinateurs 32 Bits Archimede. L'architecture ARM ne dépend pas d'un seul fabricant. Le modèle économique de l'entreprise repose sur la vente de licences à d'autres fabricants. Les SoC d'ARM se retrouvent ainsi dans la plupart des smartphones et des objets connectés.

- **L'architecture X86 :**

Voici l'architecture la plus répandu dans le monde. Conçue par Intel, elle est utilisée commercialement depuis 1978. Cette architecture a permis de développer les processeurs des ordinateurs, des serveurs ou encore de certaines tablettes. Intel Atom est la gamme SoC du célèbre fondeur. L'architecture X86 est très peu utilisée pour développer les modèles d'un système sur puce IoT.



Arm CortexM3



x386 Intel

Pour les différences exactes entre les 2 architectures, voir https://www.shiningltd.com/what-difference-between-arm-architecture-and-x86-architecture/

Exemples :

- Le Soc **Exynos 990** dispose d'un CPU 5G 8 coeurs : 2 coeurs 2.73GHz Samsung M5, deux coeurs 2.5GHz Cortex A76 et quatre coeurs 2GHz Cortex A55. Le GPU est un ARM Mali G77MP11.

- *« (…) Il y a très peu de temps, la fondation Raspberry Pi annonçait la sortie du Raspberry Pi Pico, une carte basée sur le RP2040, un SoC inédit qui intègre un processeur ARM Cortex M0+ cadencé à 133 MHz, 264 Ko de SRAM et 2 Mo de QSPI Flash pour le stockage. »*

  D'après le lien clubic

- *« (…) La chose la plus importante à comprendre sur le rôle que joue l'architecture de processeur Arm dans tout le marché de l'informatique et des communications - smartphones, PC, serveurs ou autres - est la suivante : Arm Holdings, Ltd. est propriétaire de la conception de ses puces et de l'architecture de leurs jeux d'instructions, comme le Arm64 64 bits. Conséquence, pour ses clients qui construisent des systèmes autour de ces puces, Arm a fait le plus dur du boulot. »*

  D'après le lien zdnet

2 Gestion des processus

(a) Généralités

A retenir :
Un **processus** est l'ensemble des informations correspondant à l'exécution d'une suite d'instructions par un processeur :

- la suite d'instructions elle-même, c'est à dire le *programme* qui s'exécute
- les données du programme
- les ressources (entrés-sorties, mémoire, contenu des registres processeurs, etc.) que le programme utilise

A chaque processus est associé un identifiant *unique*, son **PID** (***P**rocess **ID**entifier*) affecté par le noyau.
Le PID est un entier de 0 à 2^{15} , ou de 0 à 2^{32} sur les systèmes 64 bits.

Les processus peuvent être créés à tout moment :

- par le système lui-même (***processus système***). Le propriétaire de ces processus est le super-utilisateur *root*.
- par l'utilisateur de la machine (***processus utilisateur***)

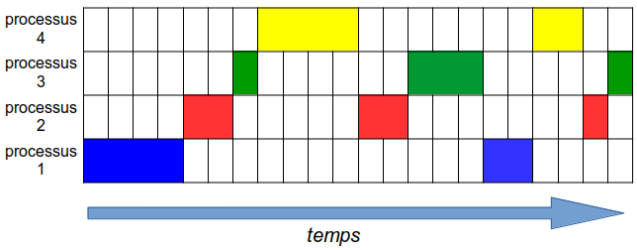
Exemple :
Le premier processus lancé au démarrage du système, et qui reste actif jusqu'à son extinction, appelé init*, reçoit le numéro 1. La tache de ce processus est de lancer les autres processus.*

Lorsqu'il a été lancé par un autre processus, appelé ***processus-père***, le ***processus-fils*** aura donc aussi un numéro de ***PPID*** (*Parent Process IDentifier*), c'est à dire le PID du processus père.
Seul le processus *init* n'a pas de processus-père, il est à la racine de la hiérarchie de tous les autres processus.

(b) Ordonnancement

A retenir :
Sur un processeur à un seul noyau ne peut s'exécuter qu'**un seul processus à la fois**.
Le noyau d'un système d'exploitation gère alors l'**ordonnancement** des processus, c'est à dire le temps-processeur et l'accès aux ressources, qu'il va leur *allouer à tour de rôle*.

L'ensemble des processus peut donc être vu comme des « sous-machines » indépendantes et ne faisant fonctionner qu'un seul programme (*mono-tache*), que le noyau feraient fonctionner à tour de rôle en passant très rapidement de l'une à l'autre pour exécuter tel ou tel travail, ce qui, pour l'utilisateur, donne l'illusion d'un système **multi-tâches**, c'est à dire exécutant plusieurs processus simultanément



Remarque : *Les processeurs à plusieurs noyaux permettent réellement d'exécuter plusieurs tâches en parallèle*).

Le ***contexte*** d'un processus, c'est à dire les zones mémoire contenant son code et ses données, les valeurs sauvegardées des registres processeurs, de la pile exécution, etc... permet de reprendre où elle en était l'exécution d'un processus qui a été interrompue par celle d'un autre.

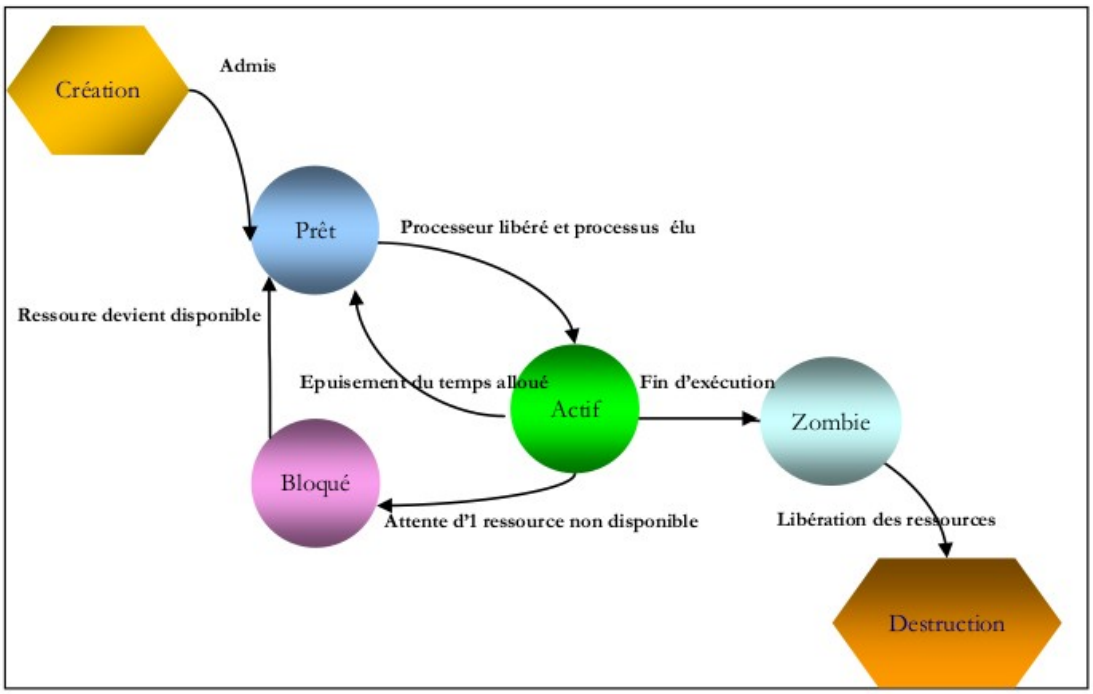
Différents algorithmes sont utilisés pour gérer l'ordonnancement des processus, et c'est toujours un domaine très actif de recherche :
Par exemple :

- **La méthode du tourniquet** (*Round Robin*): l'ordonnanceur traite la file d'attente comme une file circulaire et alloue successivement un temps processeur à chacun des processus de la file.
- **FIFO**: First In First Out
- **SJF** Shortest job first (SJF, ou SJN -Shortest Job Next-)
-

A retenir :
Un processus passe donc par plusieurs ***états*** pendant la durée où il tourne sur le système :

- ***éligible*** (prêt) : le processus est prêt à être exécuté
- ***élu*** (actif) : le processus s'exécute sur le processeur
- ***bloqué*** : en attente pendant l'exécution d'un autre processus ou d'une ressource non disponible

Un processus donné peut utiliser plus ou moins de temps-processeur ou de ressources par rapport aux autres selon la **priorité** qui lui est affecté.



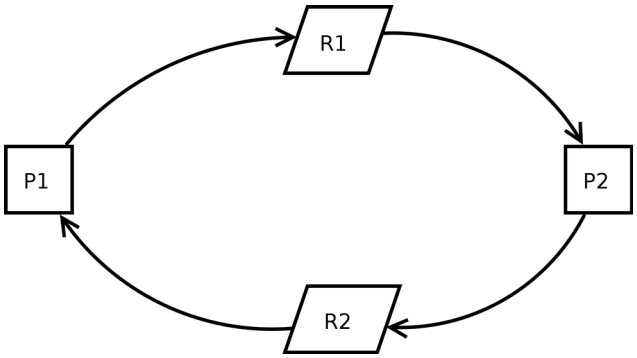
Exemple :
Sous Linux, la priorité est une valeur (appelée nice *) comprise entre 19 (la plus petite priorité) et -20 (la plus grande priorité).*
Par défaut, un processus est lancé avec une valeur nice *de 0 : un utilisateur normal peut augmenter la valeur* nice *d'un processus, mais seul le super-utilisateur* root *peut la diminuer*

(c) Interblocage

Les processus sont notés Pi et les ressources nécessaires Ri.

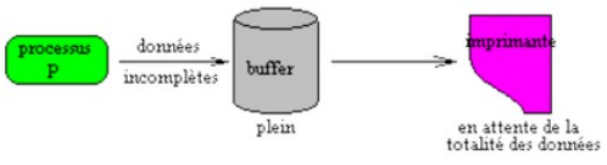
- P1 acquiert R1.
- P2 acquiert R2.
- P1 attend pour acquérir R2 (qui est détenu par P2).
- P2 attend pour acquérir R1 (qui est détenu par P1).

Dans cette situation, les deux processus sont définitivement bloqués.



A retenir :
Un **interblocage** (ou étreinte fatale, *deadlock* en anglais pour impasse) est un phénomène qui peut se produire en programmation concurrente lorsque des processus s'attendent mutuellement.

Exemple :
Le spooling massif : P veut imprimer de grandes quantités données, il les envoie donc à l'imprimante. L'imprimante utilise un spool (buffer qui créé une file de tâches), c'est à dire qu'elle attend que toutes les données à imprimer soient transférées sur le buffer avant de lancer le travail d'impression.



Le problème survient si le buffer est plein avant que P n’ai fini d’envoyer toutes ses données : P attend que le buffer se vide, et l’imprimante attend que P ait terminer le transfert...

Il existe plusieurs façons de gérer les interblocages:

- les ***ignorer*** ce qui était fait initialement par UNIX qui supposait que la fréquence des interblocages était faible et que la perte de données encourue à chaque fois est tolérable.
- les ***détecter*** : un algorithme est utilisé pour suivre l'allocation des ressources et les états des processus, il annule et redémarre un ou plusieurs processus afin de supprimer le blocage détecté.
- les ***éviter***: des algorithmes sont utilisés pour supprimer une des conditions nécessaires à la possibilité de l'interblocage

Coffman a prouvé en 1971 qu'il y a quatre conditions nécessaires pour qu'un blocage puisse avoir lieu. Voir l'article Wikipédia Conditions de Coffman pour plus de détails.