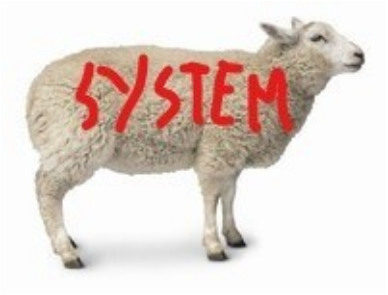
Cours – Composants intégrés d'un système sur puce



Objectifs :

- ⇨ Découvrir ce qu'est un microcontrôleur, un SoC
- ⇨ Comprendre les avantages et les inconvénients de tels systèmes
- ⇨ Mettre en œuvre une carte microcontrôleur

I – Rappels sur l'architecture d'un ordinateur

1) Principales parties

Un ordinateur est composé de plusieurs parties distinctes, reliées entre elles :

- Le ..... qui exécute les programmes et réalise les calculs
- La ..... (qui contient les données et les instructions des programmes)
- Les ..... (comme le clavier, la souris, le disque dur, la carte réseau, …)

2) La mémoire

On parle ici de la mémoire accessible directement au processeur et pas de la *mémoire de masse* comme les disques dur, SSD (*Static State Drives*), CD/DVD/Bluray ou clé USB.

Barette mémoire de type DDR (mémoire vive)

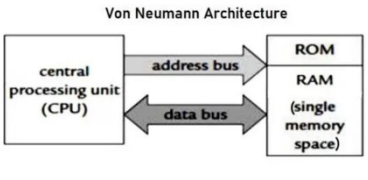
Il existe principalement deux types de mémoires :

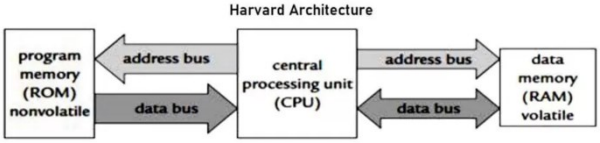
- Les mémoires accessibles en lecture/écriture (mémoire ..... ou ..... (*Random Access Memory*)). Leur nom français leur vient du fait qu'elles perdent leur contenu lorsqu'elles ne sont plus alimentées en électricité.
- Les mémoires accessibles en lecture seule (mémoire ..... ou ..... (*Read Only Memory*)). Dans cette catégorie certaines mémoire sont cependant modifiables (mais avec un accès lent et énergivore : ce sont les mémoires de type EEPROM ou FLASH). Elles gardent l'information même lorsqu'il n'y a plus d'alimentation.

3) Architectures Von Neumann et Harvard

Il existe deux façons de relier le processeur à la mémoire :

- L'architecture de ..... où la mémoire est d'un seul bloc (constitué de RAM et ROM) et où les données et les programmes sont stockés ensembles.
- L'architecture de ..... où les données et les programmes sont stockées dans deux espaces séparés ayant chacun leur canal d'accès au processeur.





Un peu d'histoire :

C'est le mathématicien américano-hongrois John von Neumann qui a élaboré en juin 1945 dans le cadre du projet EDVAC1 la première description d'un ordinateur dont le programme est stocké dans sa mémoire.
L'architecture d'Harvard a été développée dans les années 1930 à l'université d'Harvard (Etats-Unis). Les tout premiers ordinateurs ont été construits d'après ce modèle.

II - Présentation des microcontrôleurs

Les microcontrôleurs sont des puces spécialisées intégrant sur le même composant ....., ...

et ..... Ils servent dans de nombreuses applications d'automatisation comme par exemple un portail automatique, une mini station météo automatisée, un robot-aspirateur, …

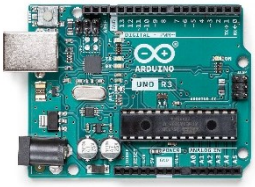
Ces microcontrôleurs possèdent généralement des processeurs de type RISC (*Restricted Instruction Set Computer*), c'est-à-dire à jeu d'instruction réduit pour simplifier leur architecture (tandis que les processeurs « classiques » sont de type CISC (*Complex Instruction Set Computer* – Jeu d'instruction complexe). Ils sont peu puissants mais intègrent de nombreuses capacités comme :

- des entrées/sorties analogique (valeurs variables) ;
- des entrées/sorties numériques (type tout ou rien) ;
- des minuteurs ;
- des bus de communication avec les périphériques.

Un microcontrôleur ne possède généralement qu'un seul programme en mémoire (situé en mémoire morte) qu'il exécute dès qu'il est sous tension et qui tourne en boucle infinie. Il n'a pas de système d'exploitation (juste quelques bibliothèques de fonctions système) et le programme accède donc directement au matériel.

1) Le microcontrôleur Arduino™

Arduino Nano
(45×18mm, 7 g)



Arduino UNO
(format carte de crédit)

Les microcontrôleurs de la gamme Arduino sont bien connus des bidouilleurs car ils sont libres (*open hardware*) et bénéficient ainsi de nombreuses déclinaisons et d'une communauté de passionnés très importante.

Application n°1 :

1) En vous aidant de la documentation technique, déterminer le type d'architecture mémoire utilisée pour Arduino UNO.

2) Donner les principales caractéristiques de son processeur principal (marque, modèle, fréquence, largeur de bus, type d'architecture…). Il sera sans doute nécessaire de consulter ses spécifications techniques (datasheet).

3) A l'aide d'une recherche internet, trouvez quelques applications utilisant un microcontrôleur Arduino™.

4) Combien coûte une carte Arduino UNO ou compatible seule (pas le kit de développement) ?

2) Avantages et inconvénients des microcontrôleurs

a. Avantages :

- ..... par intégration des différents composants, ce qui rend plus simple les applications nomades (alimentable par une simple pile).
- ..... .
- ..... car simples, sans système d'exploitation et qui ne peuvent pas être modifiés sans reprogrammation de la puce.

b. Inconvénients :

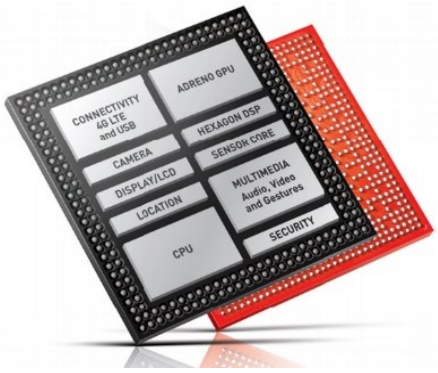
- Possèdent une ..... (peu de mémoire et de capacité de calcul)
- Ne sont ..... (pas possible de rajouter de la mémoire vive par exemple).
- ..... (si une partie ne fonctionne pas, il faut changer tout le microcontrôleur).
- Nécessitent des connaissances en électricité pour ajouter des périphériques (capteurs/actionneurs).

III – Présentation des SoC

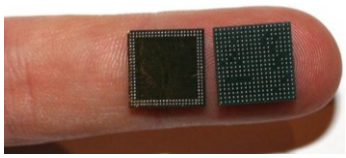
Les systèmes sur puce (*SoC* : *System On Chip* en anglais) utilisent la même philosophie que les microcontrôleurs, mais intègrent généralement des processeurs plus puissants, davantage de mémoire mais moins d'entrées/sorties. On les retrouve dans les applications nomades ou demandant de la miniaturisation comme les smartphones, dispositifs numériques (caméra, appareil photo, GPS, montre connectée, …) ou les nano-ordinateurs.

Dans un SoC, on trouve en plus du processeur des éléments parmi :

- de la mémoire vive en assez grande quantité ;
- de la mémoire flash ;
- de la mémoire cache ;
- un GPU (*Graphics Processing Unit*) ou processeur graphique ;
- un DSP (*Digital Signal Processor*) ou processeur de traitement du signal ;
- un module d'encryption/décryption ;
- un NPU (*Neural Processing Unit*) pour l'intelligence artificielle ;
- un modem ;
- une carte réseau ;
- une carte de connectivité wifi/bluetooth/4G/5G ;
- un GPS ;
- des modules d'entrée-sortie ;
- …



Il s'agit pratiquement d'intégrer tous les composants d'un micro-ordinateur sur une puce unique et dans un espace très réduit.



SoC BCM2835
(présent dans les Raspberry Pi)

Pour maîtriser leur consommation d'énergie, les processeurs des SoC sont capables de fonctionner à des fréquences variables ou de désactiver certains cœurs pour économiser l'énergie lorsque la charge de calcul n'est pas trop grande.

L'architecture mémoire est généralement de type Von Neumann pour offrir davantage de flexibilité.

Il existe plusieurs types de systèmes sur puces :

- Les systèmes **basés sur un microcontrôleur** qui possèdent des capacités étendues d'entrées/sorties.
- Les systèmes **basés sur un microprocesseur** qui ont une grande puissance de calcul.
- Les systèmes **spécialisés** qui intègrent des modules spécifiques.
- Les systèmes **programmables** qui intègrent des circuits logiques programmables comme des FPGA[1] ce qui permet de programmer certaines de leurs fonctionnalités.

Tous ces systèmes sont suffisamment puissants pour faire tourner des systèmes d'exploitation. C'est généralement linux et ses variantes (comme Android) qui sont choisis car ils utilisent moins de puissance et ne nécessitent pas de licence payante.

Avec les SoC on a à peu près les mêmes avantages et inconvénients que les microcontrôleurs (sauf la puissance de calcul qui peut être assez satisfaisante dans le cas des SoC). On les retrouve donc logiquement dans la plupart des systèmes embarqués.

Application n°2 :

1) En vous aidant d'une application comme CPU Info, déterminer le SoC utilisé sur votre smartphone. Si vous n'en avez pas, choisissez un modèle et trouver dans sa documentation le modèle de SoC qu'il contient, son nombre de cœurs, sa vitesse d'horloge et sa température de fonctionnement actuelle.

2) En utilisant les informations du site www.raspberrypi.com déterminer les caractéristiques du SoC des Raspeberry Pi 5 (Modèle, architecture, nombre de cœur, fréquence, RAM, …).

Carte Raspberry Pi 5

3) Quel est le prix d'un Raspberry Pi 5 avec 4 Gio de mémoire vive ? Comparer au prix d'un ordinateur.

IV - Mise en œuvre de la carte microbit™

Pour conclure cette séquence, nous allons mettre en œuvre une carte pouvant être programmée directement en python : la carte Microbit.

Application n°3 :

1) Examinez la documentation de la carte microbit sur le site microbit.org. La carte dont nous disposons est une microbit « original », la version 1. S'agit-il d'une carte de type microcontrôleur ou SoC ?

Pour programmer la carte microbit, nous utiliserons Thonny en mode « BBC microbit » (à changer dans le menu Exécuter/Configurer l'interpréteur). Lorsqu'on exécute le programme dans Thonny, celui-ci va transférer le programme sur la carte microbit (le « flasher ») et la ré-initialiser pour lancer le programme.

2) Copier-coller le programme très simple ci-contre et exécutez-le (le fait de lancer le programme dans Thonny va le flasher sur la microbit et relancer celle-ci. Vérifier qu'il fonctionne correctement.

```
from microbit import *

display.scroll("Microbit !")
print("Ceci s'affiche dans la console")
```

[1] *Field-Programmable Gate Array* : Composant permettant de configurer par programmation un circuit logique complet.

La documentation complète se trouve sur https://microbit-micropython.readthedocs.io/fr/latest/

3) Ecrire un programme « pixels » pour la microbit qui allume successivement chacune des leds de l'afficheur, puis qui les éteigne successivement et boucle ainsi à l'infini. Pour les plus avancés, vous pouvez faire en sorte que les leds s'allument et s'éteignent en spirale.

4) Ecrire un autre programme « equilibre » qui allume une unique led au centre de l'afficheur. Ce point lumineux doit se déplacer dans les 4 directions en fonction de l'inclinaison de la carte : il va à droite si la carte penche vers la droite, à gauche si elle penche à gauche… Le point reste bloqué sur le bord même si la carte reste inclinée du même côté.

Le programme suivant va utiliser le module `radio` de la microbit. Afin que les cartes puissent communiquer, il faut qu'elles soient toutes sur le même canal. Nous utiliserons le canal par défaut.

5) Ecrire un programme « envoi_simple » qui envoie le message `"Message A"` si on appuie sur le bouton A et `"Message B"` si on appuie sur B. Le programme affiche en boucle tous les messages reçus provenant des autres cartes. Vous pouvez personnaliser les messages envoyés avec A et B pour différentier les cartes.

Pour le programme suivant, la fonction `ticks_ms` du module `time` pourra être utile.

6) Ecrire un programme « reflexe » qui attend sur la radio un message `"A"` ou `"B"`. Dès qu'il reçoit le message, il affiche la lettre correspondante sur l'écran et compte le temps écoulé entre l'affichage de la lettre et l'appui sur le bouton concerné (si le bouton était déjà appuyé au moment de l'affichage de la lettre, il doit d'abord être relâché avant qu'on puisse comptabiliser son appui). La carte efface l'affichage puis envoie alors via le module radio le temps en milliseconde mis pour appuyer suivi du nom du programmeur séparés par une virgule.

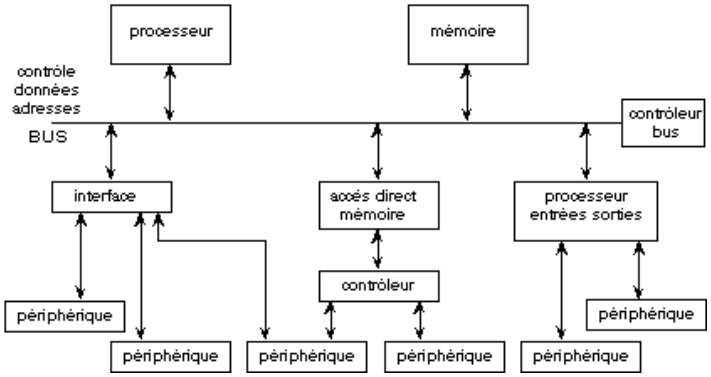
7) (pour les plus rapides) Ecrire un programme « cache-cache » qui attend que l'utilisateur appuie sur A ou B. Si l'utilisateur appuie sur A, la carte passe en mode caché et envoie en boucle le message `"caché !"` toutes les 250 ms jusqu'à l'appui sur B. Si l'utilisateur appuie sur B, la carte passe en mode recherche et fait clignoter l'écran plus ou moins vite en fonction de la distance (puissance du signal) la séparant de la carte envoyant le message `"caché !"`. La carte reste en mode recherche jusqu'à ce qu'on appuie sur A où elle revient au départ (choix du mode).

Références :

Cours complet sur les SoC : http://www.monlyceenumerique.fr/nsi_terminale/arse/a1_systeme_%20sur%20_puce.html
Différence RISC/CISC : https://www.geeksforgeeks.org/computer-organization-risc-and-cisc/
Description d'un FPGA : https://www.futura-sciences.com/tech/definitions/technologie-fpga-8700/
Ressources sur la BBC Microbit : https://github.com/mcauser/awesome-microbit

Exercices – Composants intégrés d'un système sur puce

Exercice 1 : Accès direct à la mémoire



On considère le schéma ci-contre représentant l'architecture interne d'un système informatique.

1) Quel est le type d'architecture mémoire présenté sur ce schéma ?
2) Qu'est-ce que l'« accès direct à la mémoire » (DMA : Direct Memory Acces en anglais) représenté sur ce schéma ? Faire une recherche et expliquer le fonctionnement et l'intérêt de ce système. Est-il utilisé sur les ordinateurs ? Dans les micro-contrôleurs ?

Exercice 2 : Harvard vs Von Neumann

Compléter le tableau suivant comparant les caractéristiques, avantages et inconvénients des deux architectures mémoire.

Aspects	**Architecture Von Neumann**	**Architecture Harvard**
Organisation mémoire	Mémoire unique pour les données et les programmes	
Accès à la mémoire		Le processeur possède un bus d'accès séparé et indépendant pour chaque type de mémoire.
Performance	Performances dégradées (goulot d'étranglement) si de nombreux périphériques accèdent simultanément à la mémoire.	
Modification des programmes		Difficile de modifier les programmes ou d'avoir des programmes qui modifient leur propre code.
Applications	Adapté aux ordinateurs non spécialisés où la flexibilité est importante.	

Exercice 3 : Où sont les SoC ?

Parmi les schémas suivants, lesquels correspondent à un SoC ou a un microcontrôleur ?

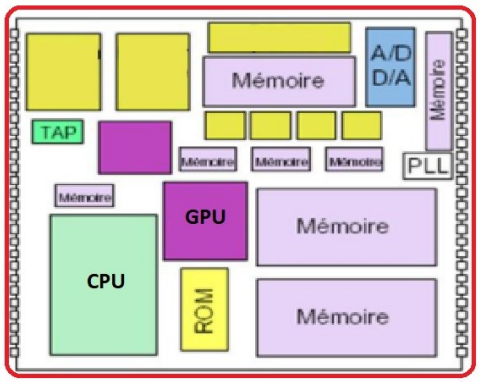


Schéma 1

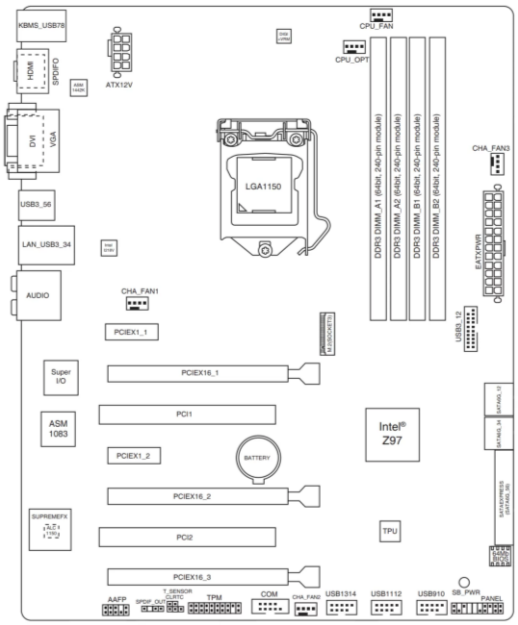


Schéma 2

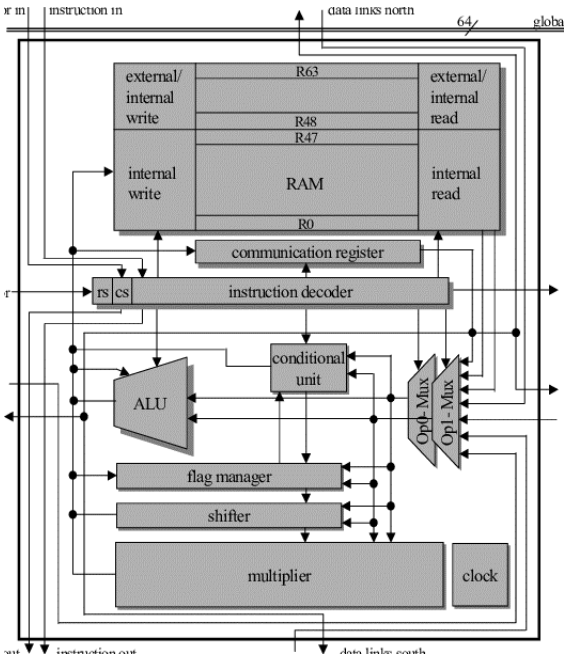


Schéma 3

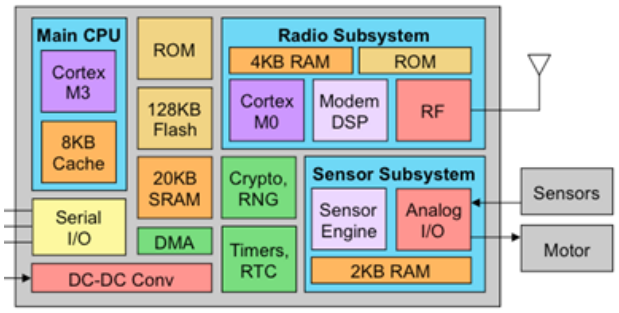


Schéma 4

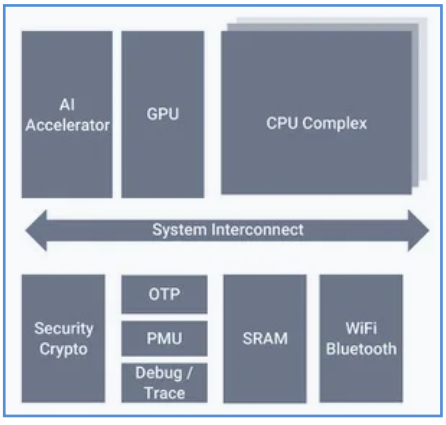


Schéma 5

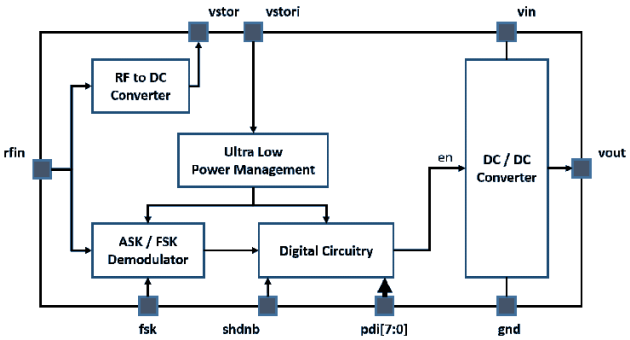


Schéma 6