Système à Micro-contrôleur

Présenté par

Mr. Hicham MEGNAFI
Email : hicham.megnafi@gmail.com
ESSA Tlemcen, Algeria

1er année – spécialité - Automatique

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Plan de Cours

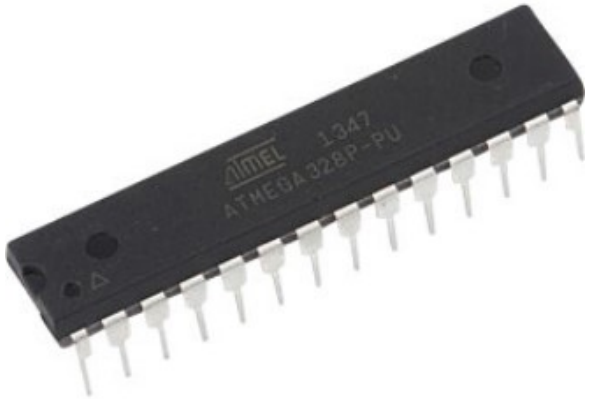
1. **Projets réalisés**
2. **Introduction**
3. **Types d'architecture**
4. **Les Processeurs**
5. **Type des mémoires**
6. **Les Bus de communication**
   - Définition
   - Principe de communication
   - Les différents types de bus (internes/externes)
7. **Les Entrés/ Sorties**
   - Introduction
   - Transmission série/ parallèle
   - Transmission synchrones /asynchrones
   - Le protocole
   - Normes RS-232 et DB 25
8. **Les interruptions**
   - Présentation
   - Déroulement d'une interruption
     - Déroulement d'une interruption
     - Interrupt Service Routine
     - Interruptions imbriquées et priorités
     - Contrôleur d'interruptions
   - Les types d'interruption
     - Interruption système
     - Exceptions
     - Interruptions matérielles
     - Interruptions logicielles

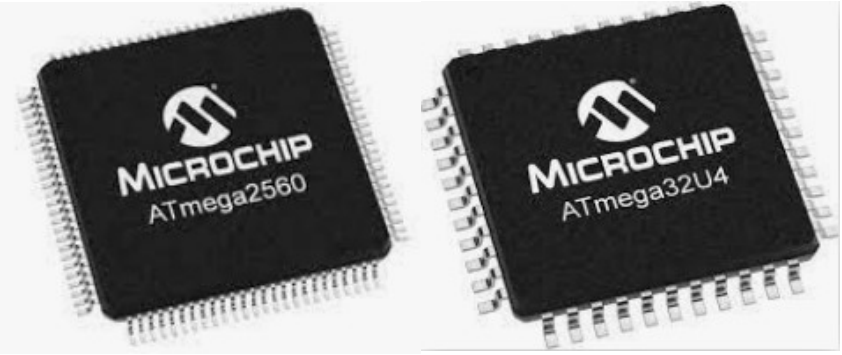
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Plan de Cours (2/2)

9. Microcontrôleur PIC16F84A

- Présentation générale du **PIC16F84A**
- Architecture du **PIC16F84A**
- Brochage et caractéristiques principales
- Structure interne
- Principe de fonctionnement du **PIC16F84A**
- La mémoire de programme
- La mémoire de données (RAM)
- Les registres
- Les ports d'entrées/sorties
- Déroulement d'un programme
- Mise en oeuvre





Plan de Cours

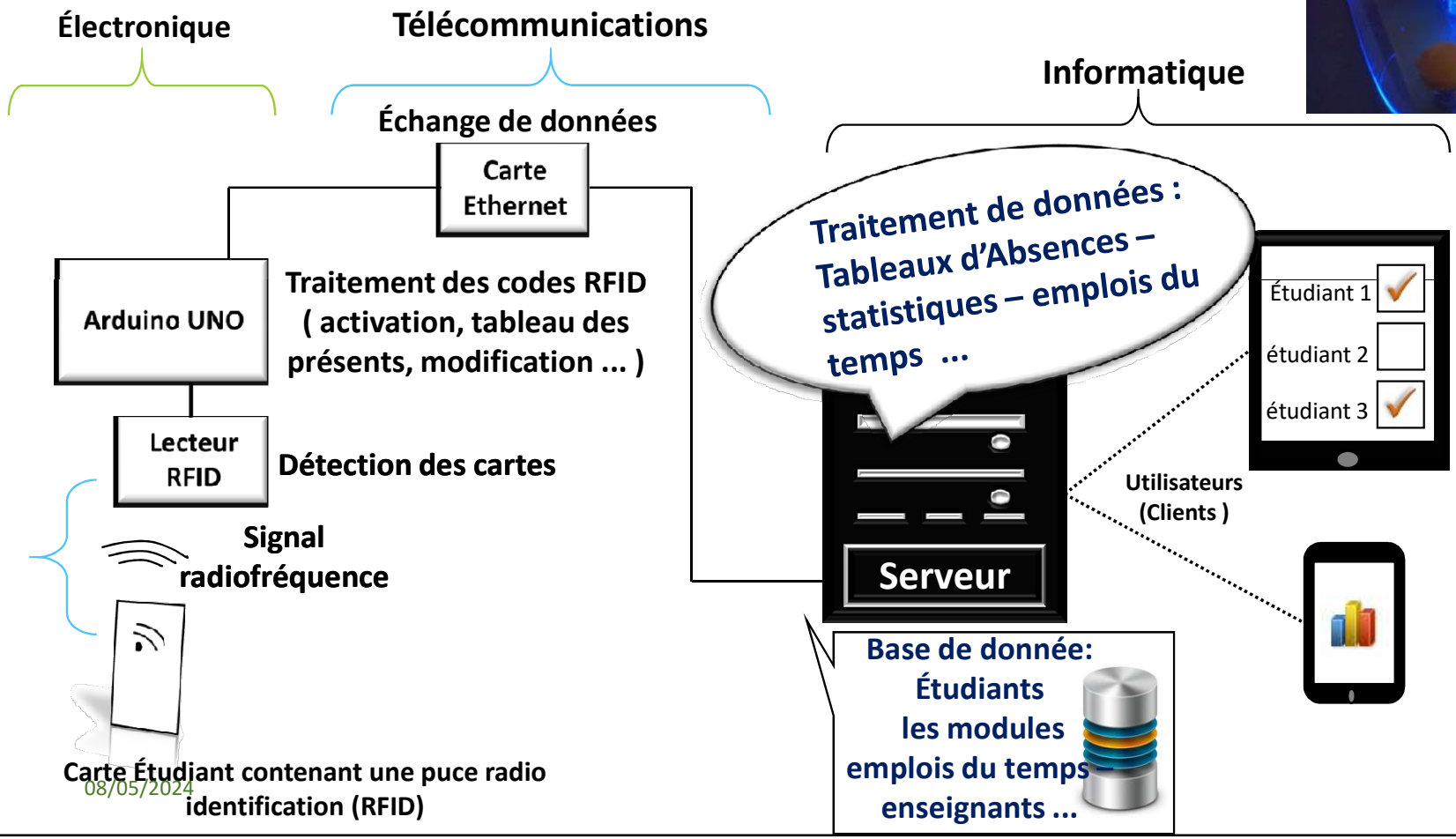
1. ***Projets réalisés***
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

1. *L'utilisation de microcontrôleur Motorola 68HC11-E9*
2. *Réalisation d'une plaque électronique qui permet d'envoyer les mesures d'une tension analogique a un micro-ordinateur via RS232 (PIC and CAN)*
3. *Système automatique de gestion de présence*
4. *Configuration et assemblage d'un drone quad rotor*
5. *Réalisation d'une imprimante 3D*

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

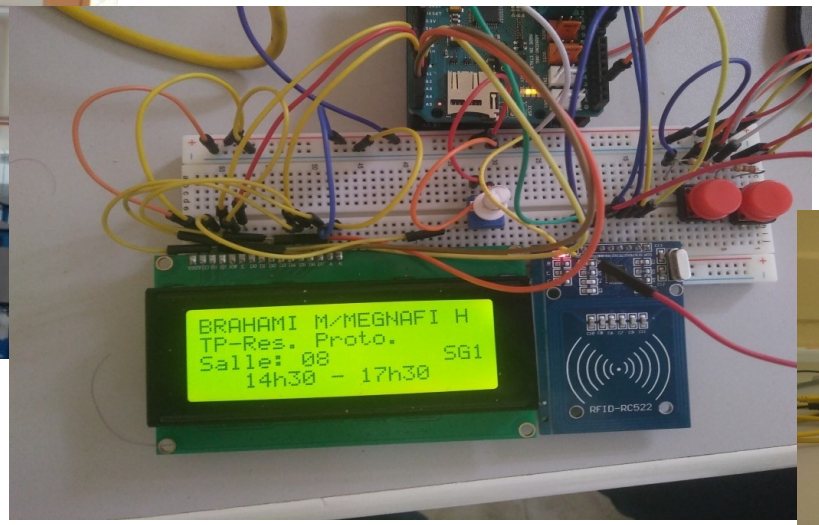
1. Un système automatique de gestion de présence (1/5)

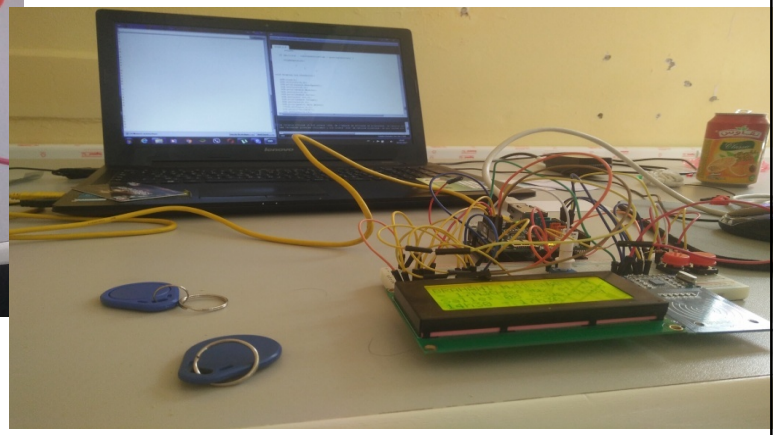


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

1. Un système automatique de gestion de présence (2/5)



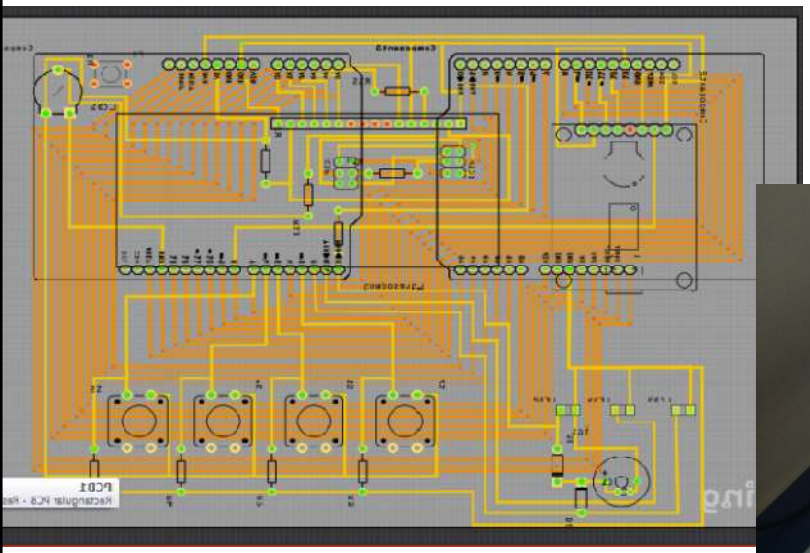


1. Premier prototype sur une plaque d'essai, après une version quasi complète du programme ...

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

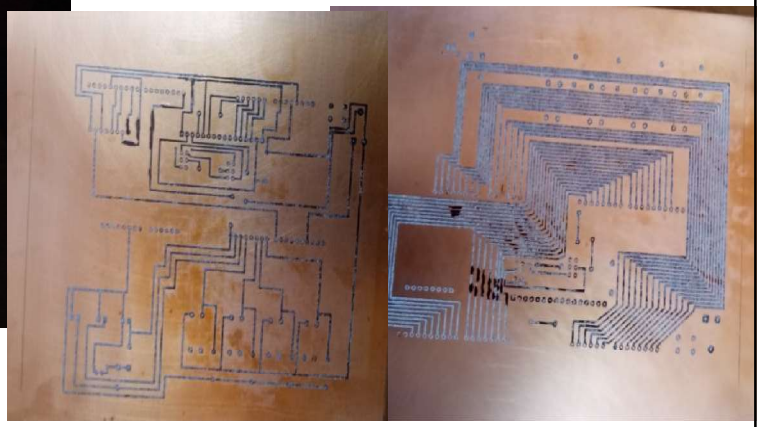
Projets réalisés

1. Un système automatique de gestion de présence (3/5)



2. Conception assisté par ordinateur

➔ Logiciel utilisé

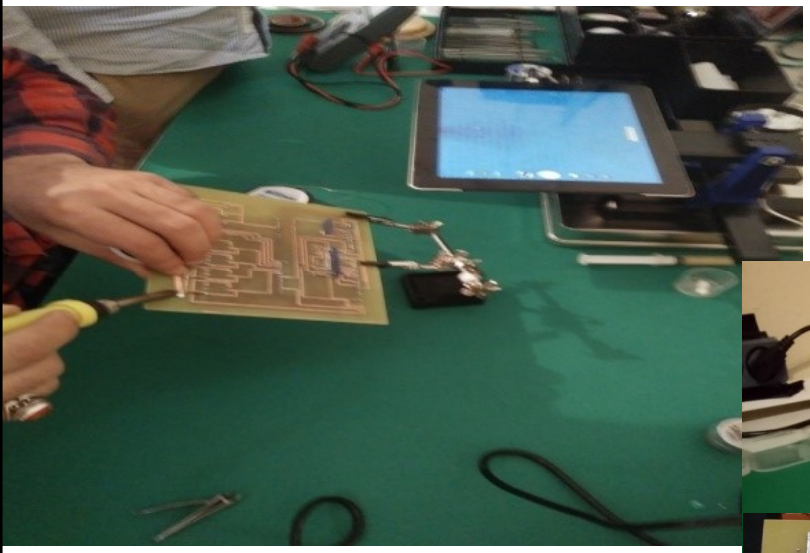


3. Réalisation du circuit imprimé à double face

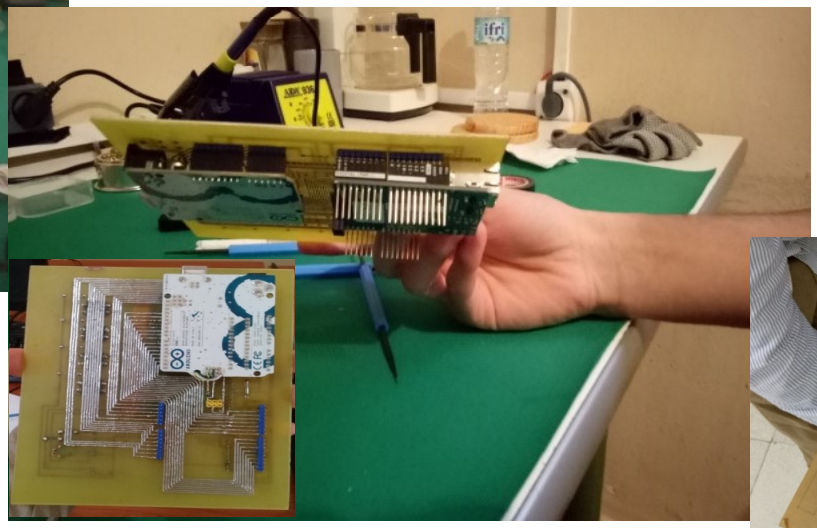
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

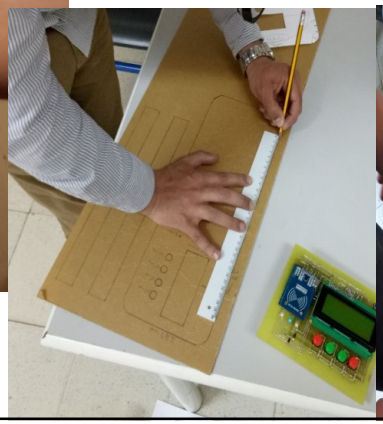
Projets réalisés

1. Un système automatique de gestion de présence (4/5)



4- Implémentation des composons électroniques sur le circuit imprimé à double face et la réalisation de boitier ...

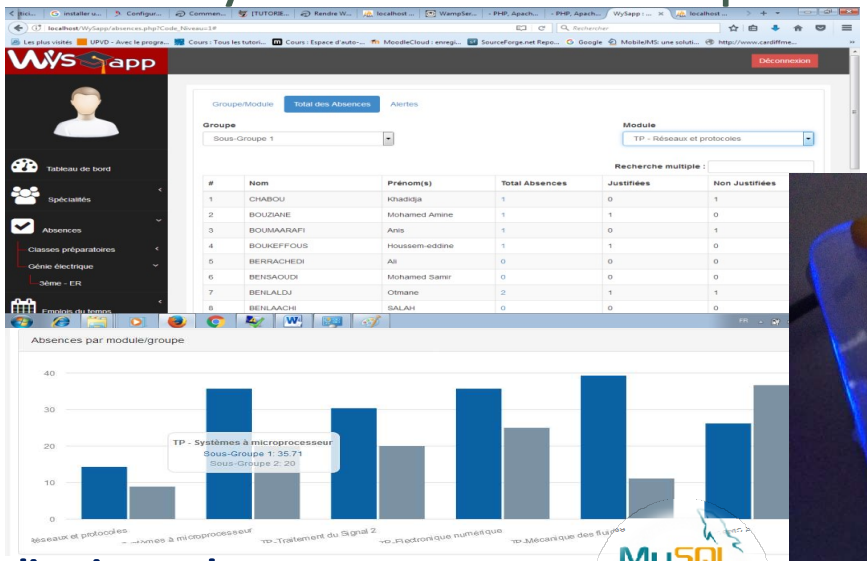


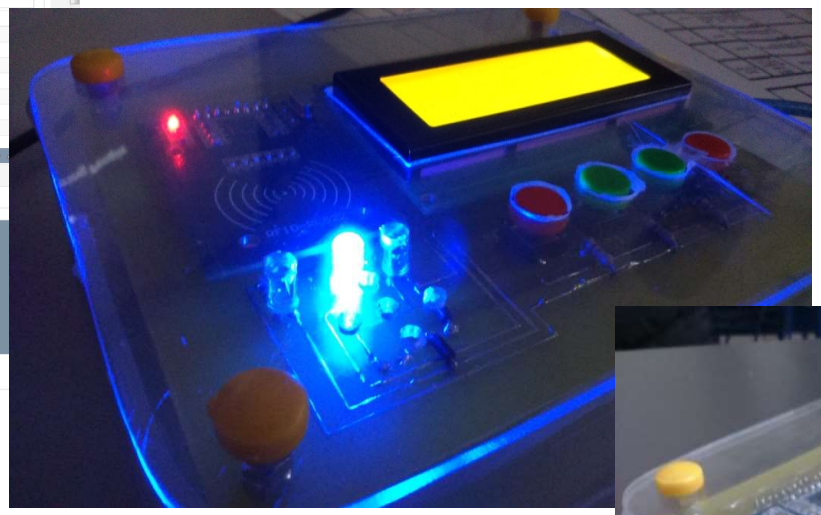


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

1. Un système automatique de gestion de présence (5/5)





L'application web...

1- Conception de la base de données

2- Développement des programmes de :

Requêtes ➔ MySQL

Traitement ➔ PHP

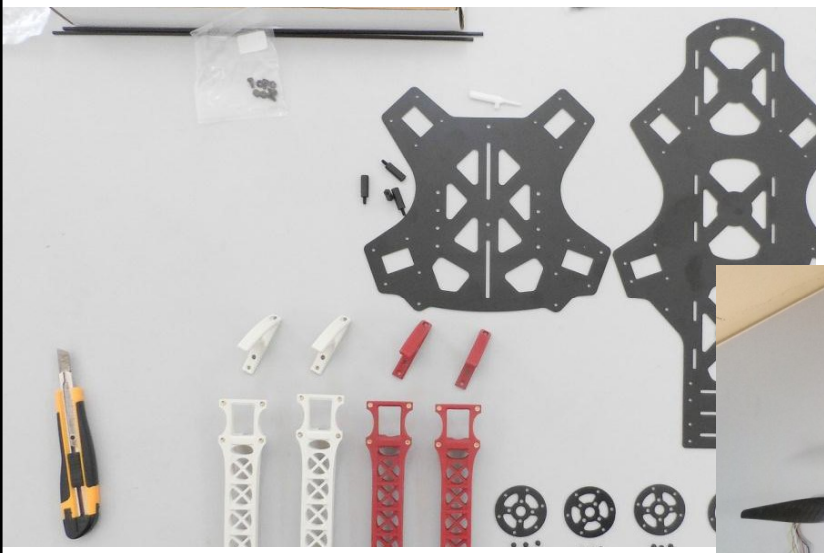
3- une conception des interfaces utilisateurs multiplateforme

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

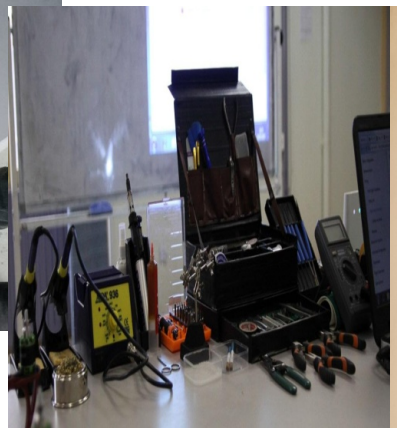
Projets réalisés

2. Configuration et assemblage d'un drone quadrotor (1/

1. Assemblage hors-châssis



Chassis du quadrotor 450 mm



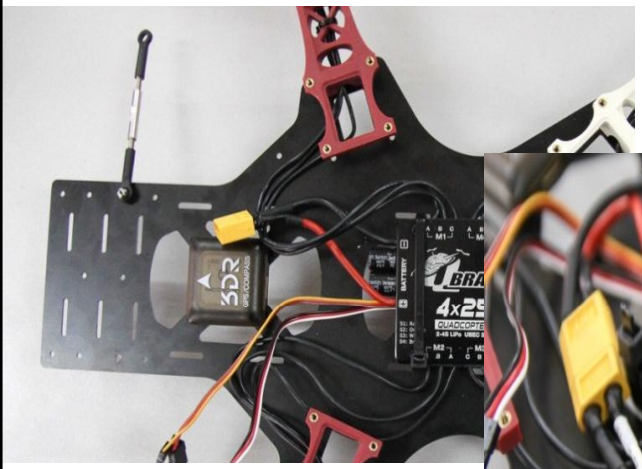
Outils d'assemblage

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

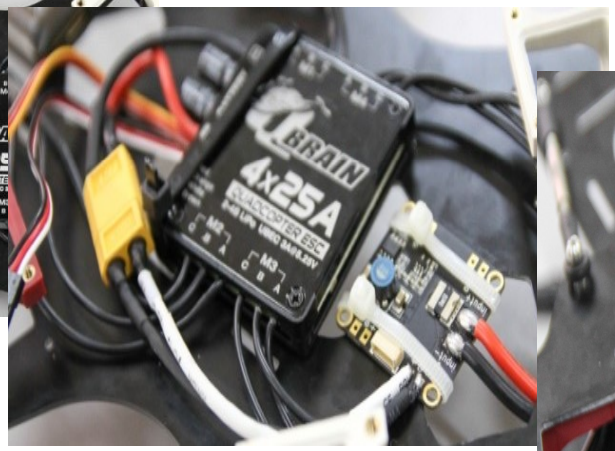
Projets réalisés

2. Configuration et assemblage d'un drone quadrotor (2/

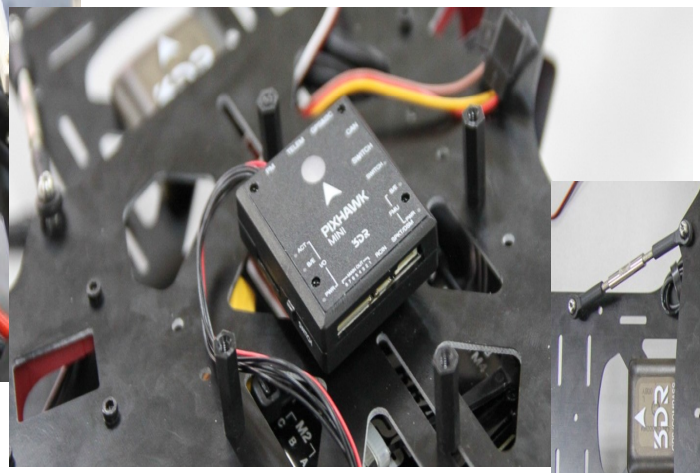
2. Assemblage complet avec les composants électroniques



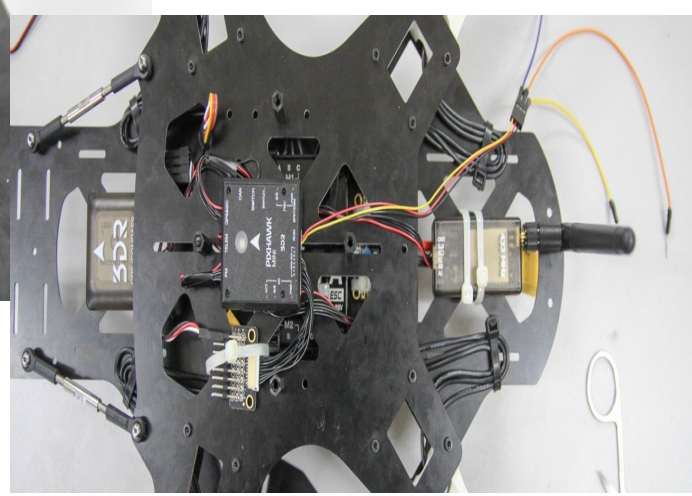
*Montage de l'ESC (*Contrôleur électronique de vitesse*)*



Montage du module de puissance



Montage du contrôleur de vol Pixhawk Mini

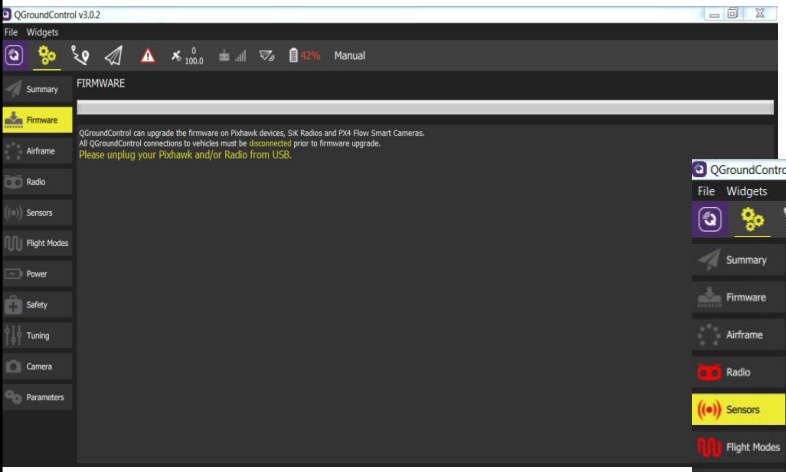


Montage du module GPS et de la télémétrie

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

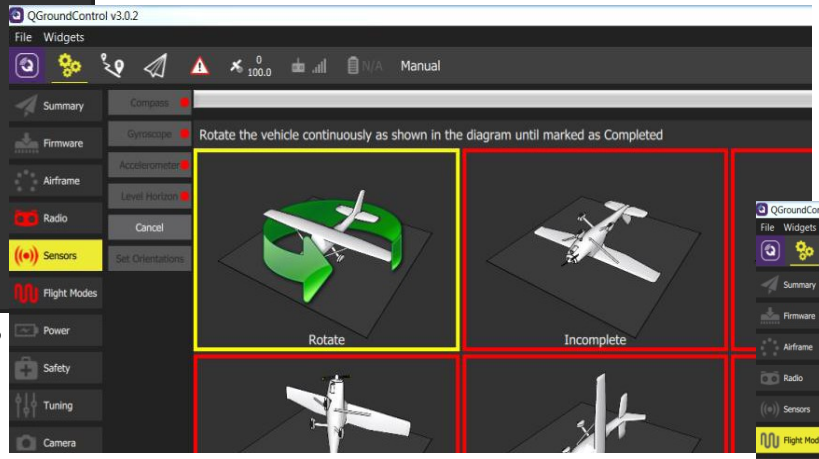
Projets réalisés

2. Configuration et assemblage d'un drone quadrotor (3/

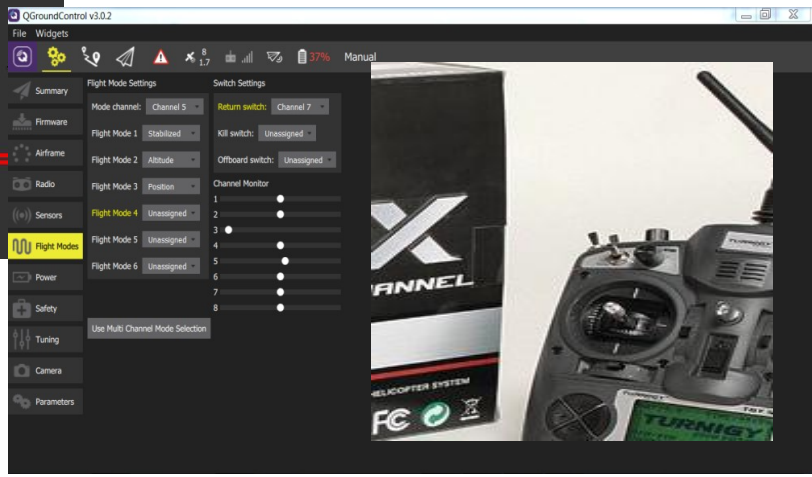


implémentation du firmware autopilote

3. Installation et configuration de la station de base



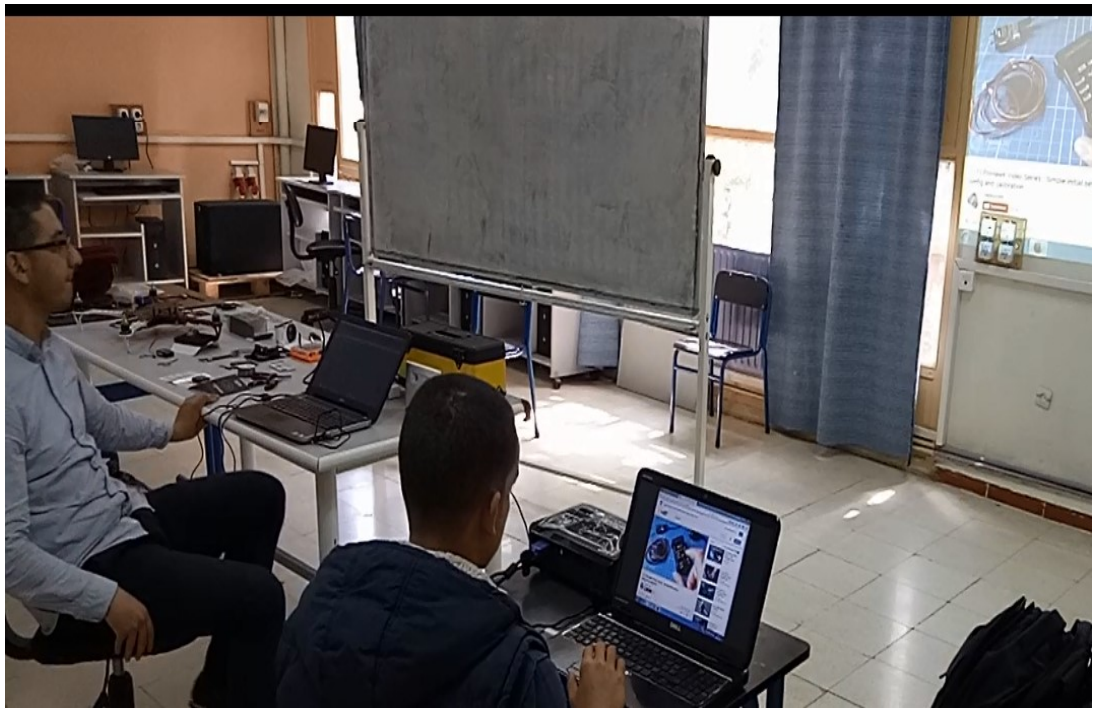
Calibrage du récepteur radio



Configuration des modes de vol

Projets réalisés

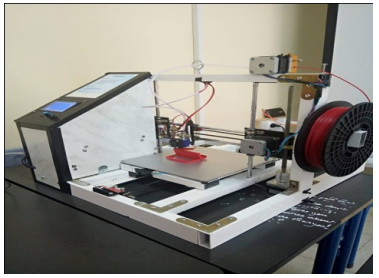
2. Configuration et assemblage d'un drone quadrotor (4/

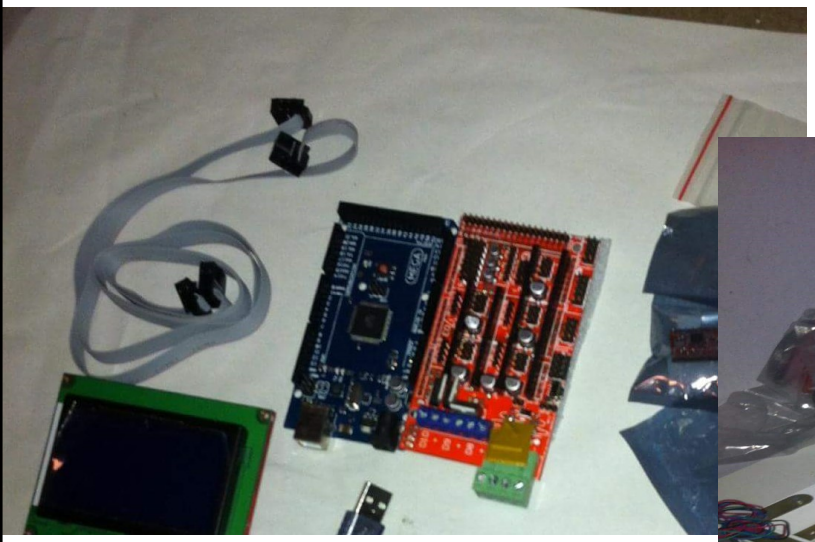


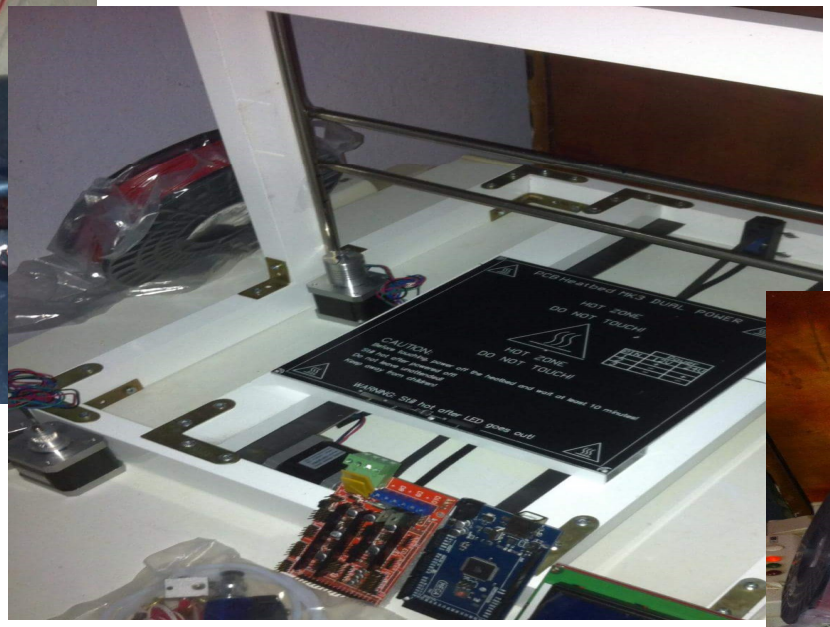
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

3. Réalisation d'une imprimante 3D (Arduino) (1/2)

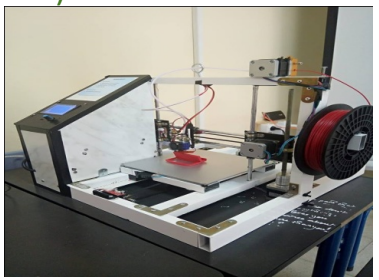


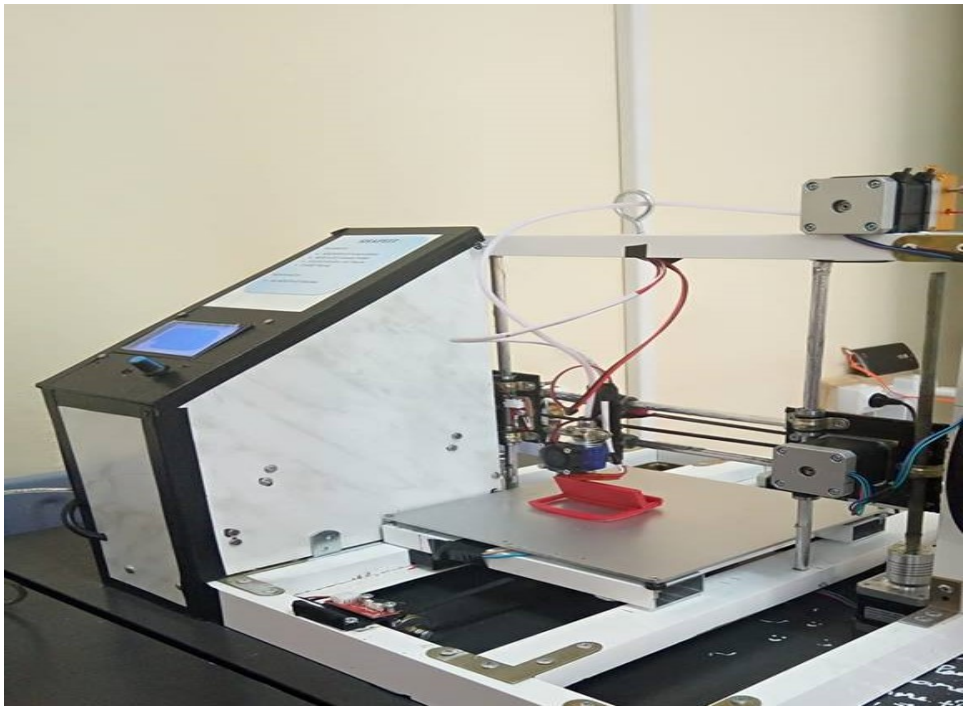


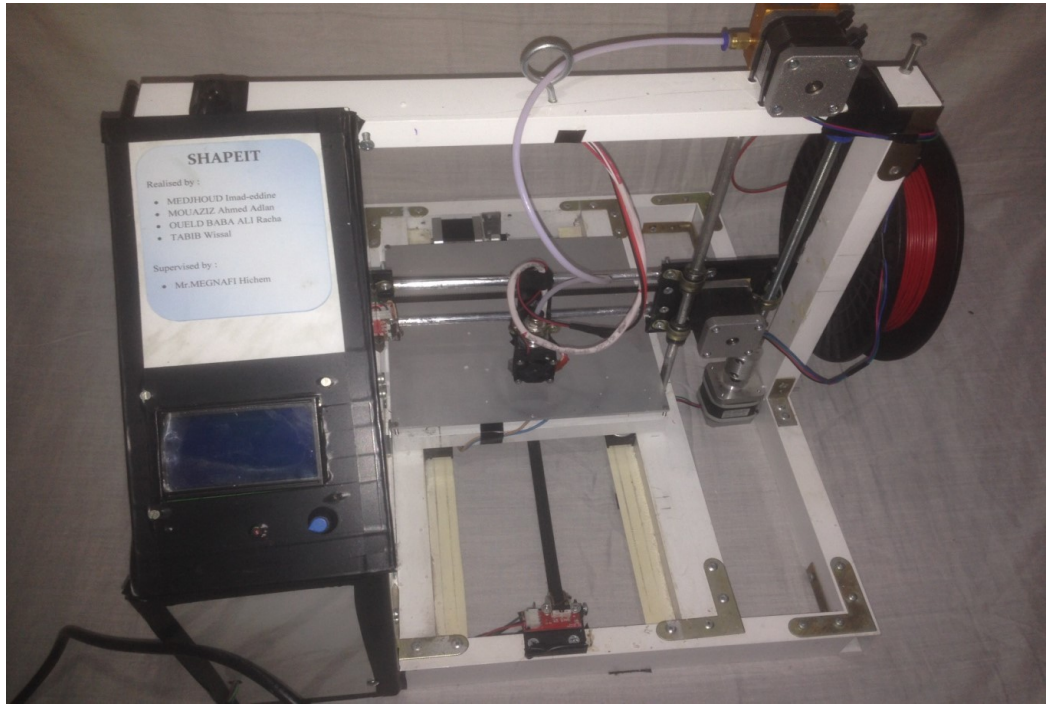


Projets réalisés

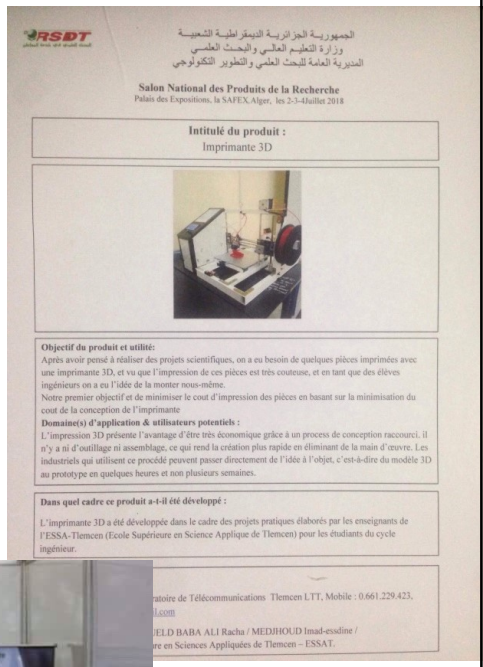
3. Réalisation d'une imprimante 3D (Arduino) (2/2)







Mr. MEGNAFI Hicham (ESSA-Tlemcen)



الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المديرية العامة للبحث العلمي والتطوير التكنولوجي

Salon National des Produits de la Recherche
Palais des Expositions, la SAFEX Alger, les 2-3-4Juillet 2018

Intitulé du produit :
Imprimante 3D

Objectif du produit et utilité:
Après avoir pensé à réaliser des projets scientifiques, on a eu besoin de quelques pièces imprimées avec une imprimante 3D, et vu que l'impression de ces pièces est très couteuse, et en tant que des élèves ingénieurs on a eu l'idée de la monter nous-même.
Notre premier objectif et de minimiser le cout d'impression des pièces en basant sur la minimisation du cout de la conception de l'imprimante

Domaine(s) d'application & utilisateurs potentiels :
L'impression 3D présente l'avantage d'être très économique grâce à un process de conception raccourci. il n'y a ni d'outillage ni assemblage, ce qui rend la création plus rapide en éliminant de la main d'œuvre. Les industriels qui utilisent ce procédé peuvent passer directement de l'idée à l'objet, c'est-à-dire du modèle 3D au prototype en quelques heures et non plusieurs semaines.

Dans quel cadre ce produit a-t-il été développé :

L'imprimante 3D a été développée dans le cadre des projets pratiques élaborés par les enseignants de l'ESSA-Tlemcen (Ecole Supérieure en Science Applique de Tlemcen) pour les étudiants du cycle ingénieur.

atoire de Télécommunications Tlemcen LTT, Mobile : 0.661.229.423,
l.com
ELD BABA ALI Racha / MEDJHOUD Imad-essdine /
re en Sciences Appliquées de Tlemcen – ESSAT.

Salon National des Produits de la R
Palais des Expositions SAFEX-A
Du 2 au 4 Juillet 2018

IMPRIMANTE 3D

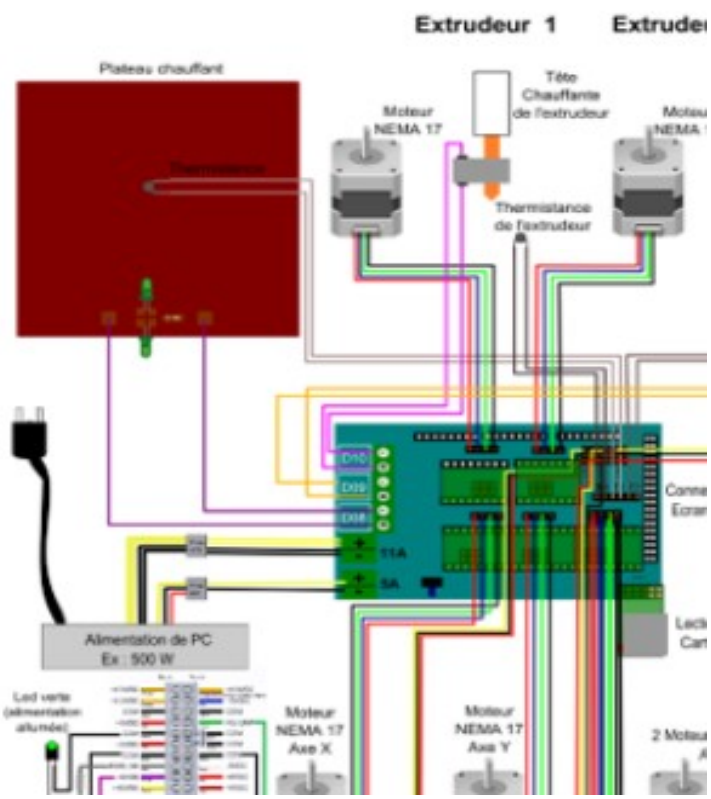
Description du produit

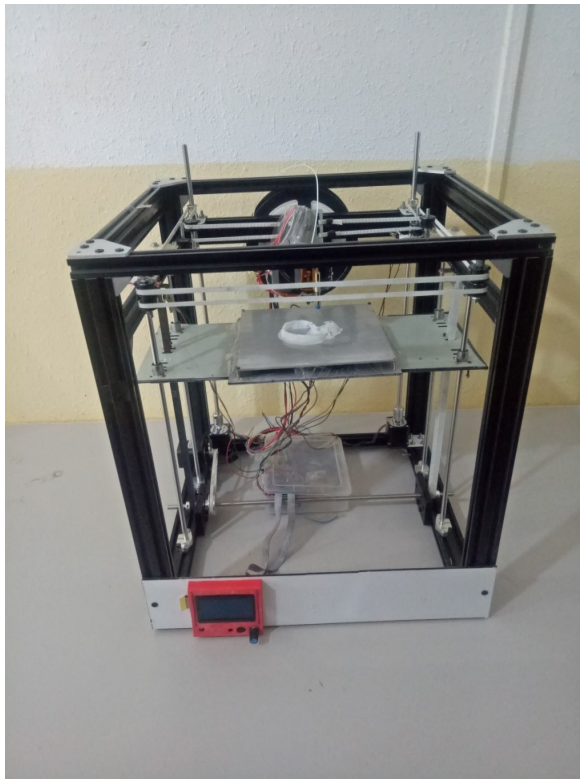
Après avoir pensé à réaliser des projets scientifiques, on a
quelques pièces imprimées avec une imprimante 3D
l'impression de ces pièces est très couteuse, et en tant q
ingénieurs on a eu l'idée de la monter nous-même.
Notre premier objectif et de minimiser le cout d'impression
basant sur la minimisation du cout de la conception de l'imp

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

4. Réalisation d'une imprimante 3D Version 2

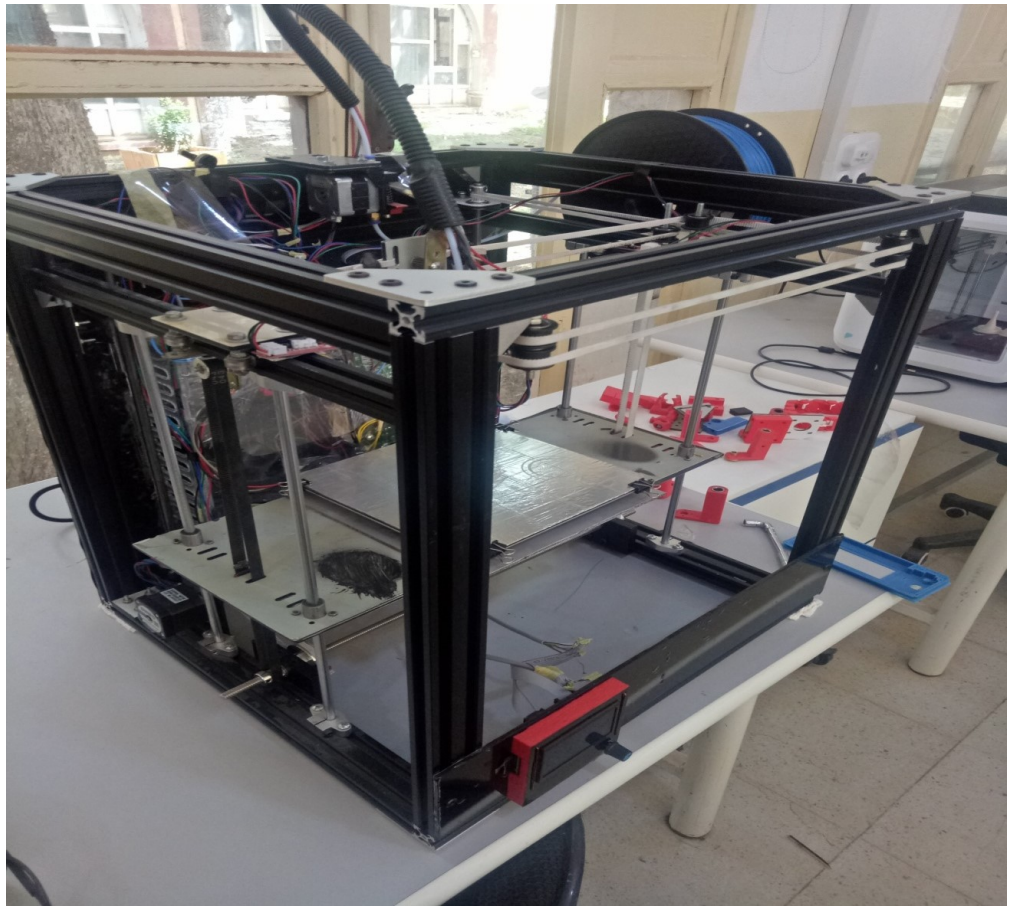


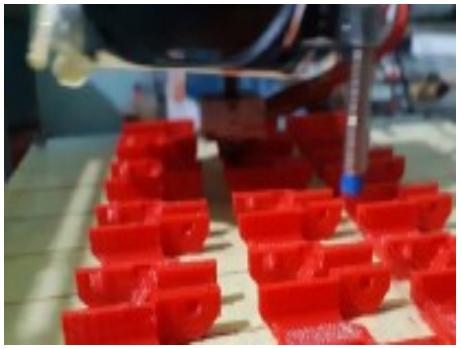


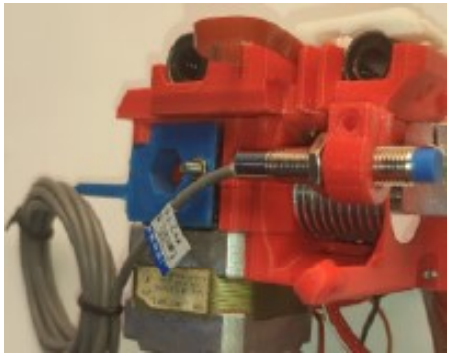
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

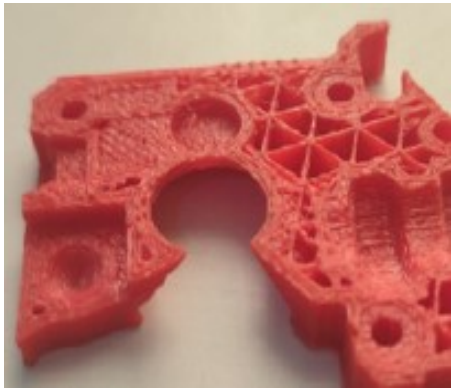
Projets réalisés

4. Réalisation d'une imprimante 3D Version 2









Mr. MEGNAFI Hicham (ESSA-Tlemcen)

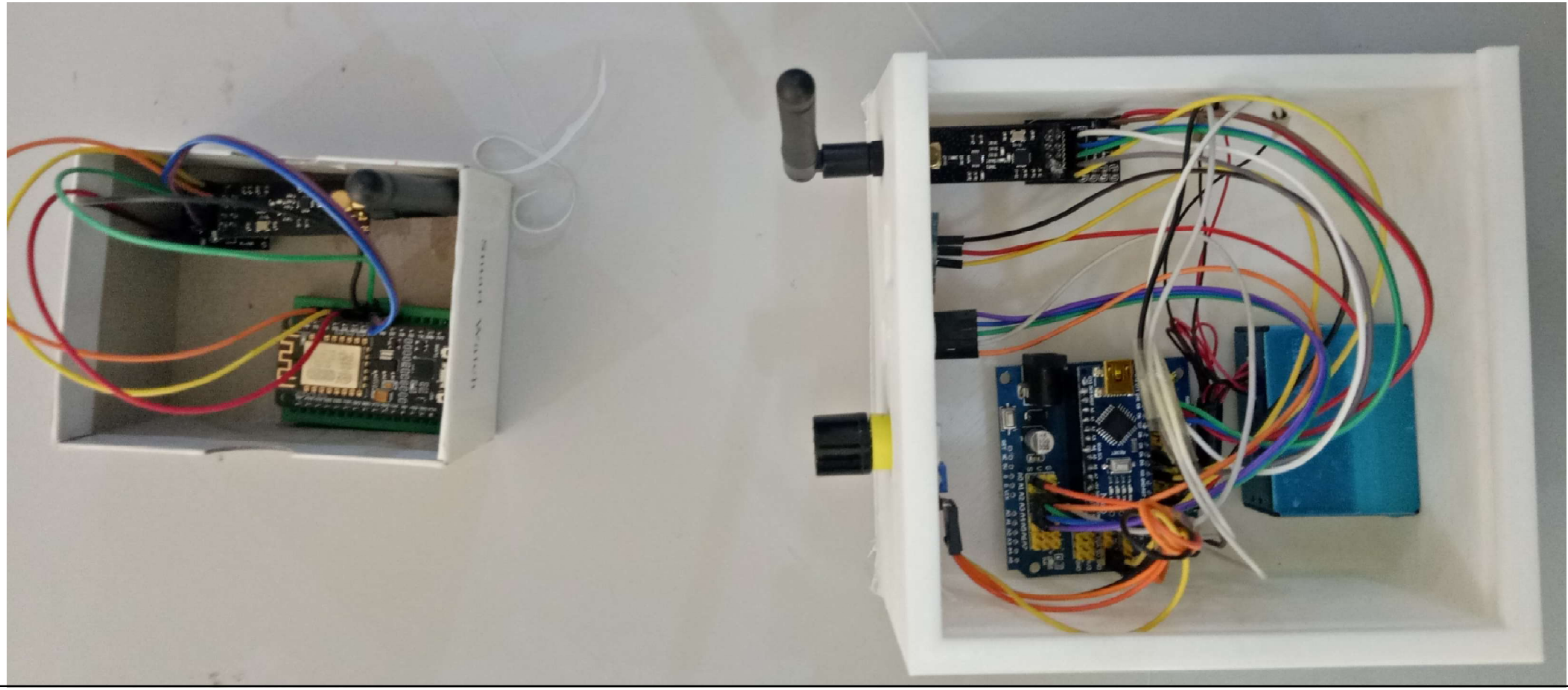
Projets réalisés

5. Drone pour la surveillance de la qualité de l'air

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

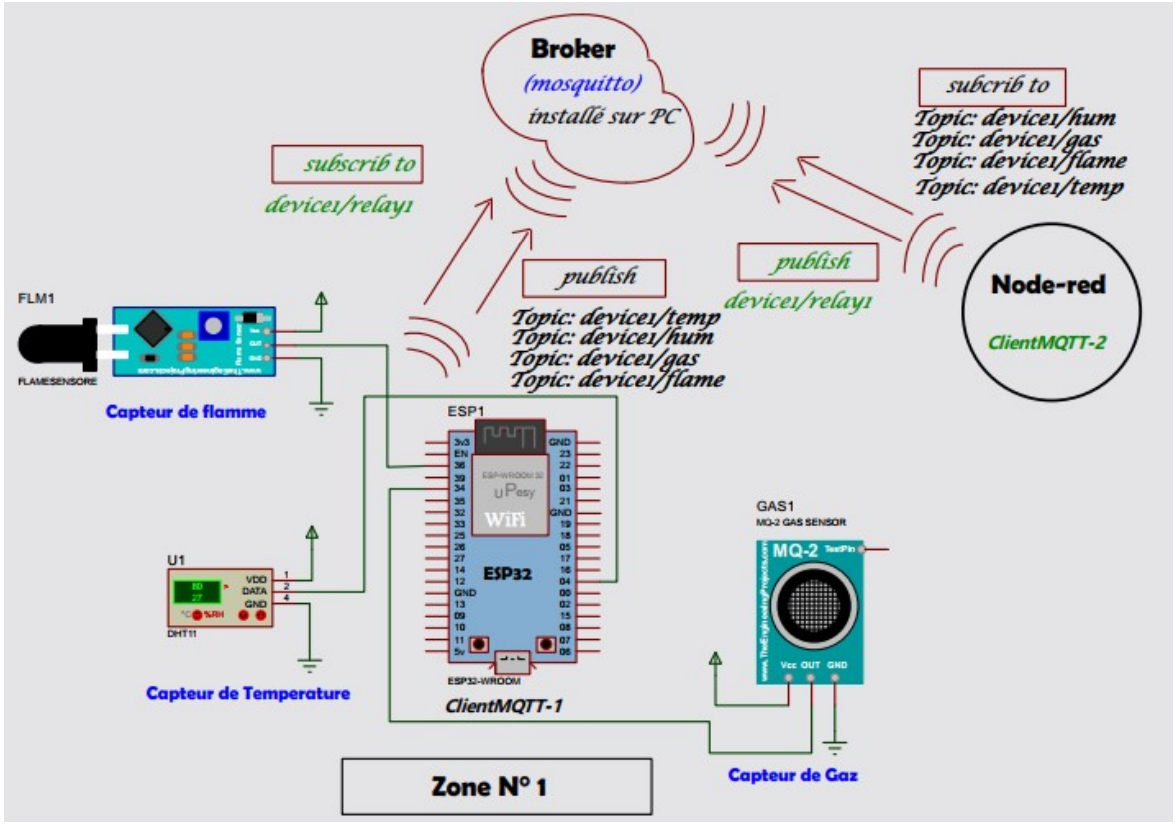
5. Drone pour la surveillance de la qualité de l'air

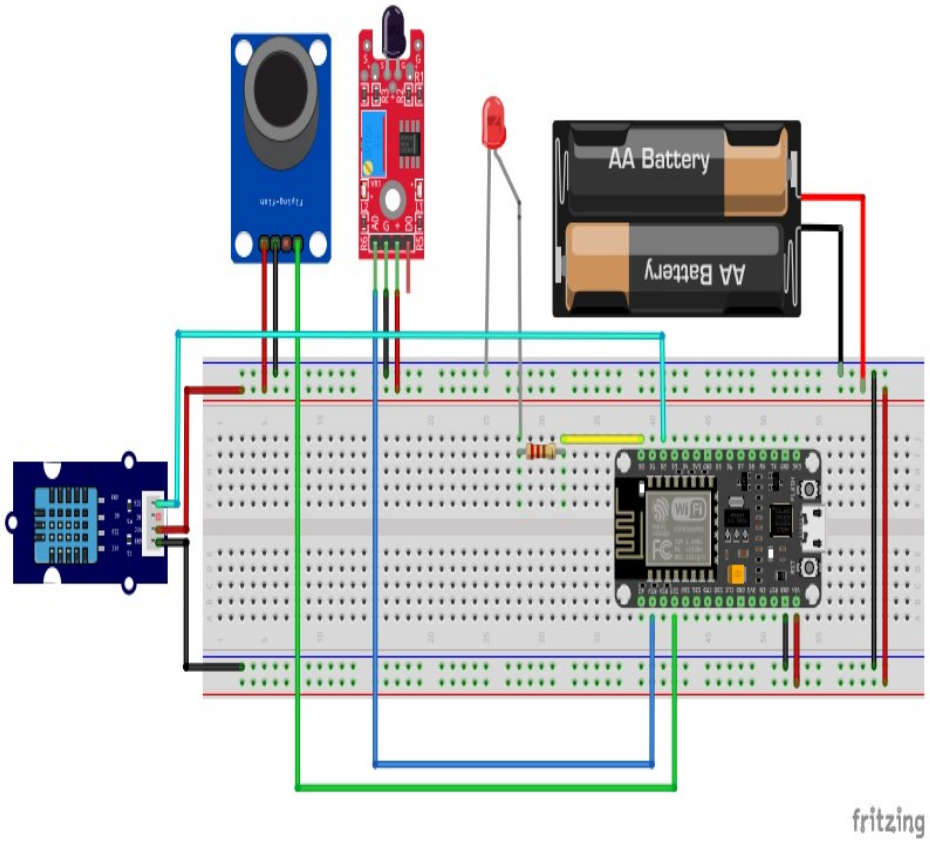


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

6. Réalisation d'un système embarqué pour la sécurité incendie

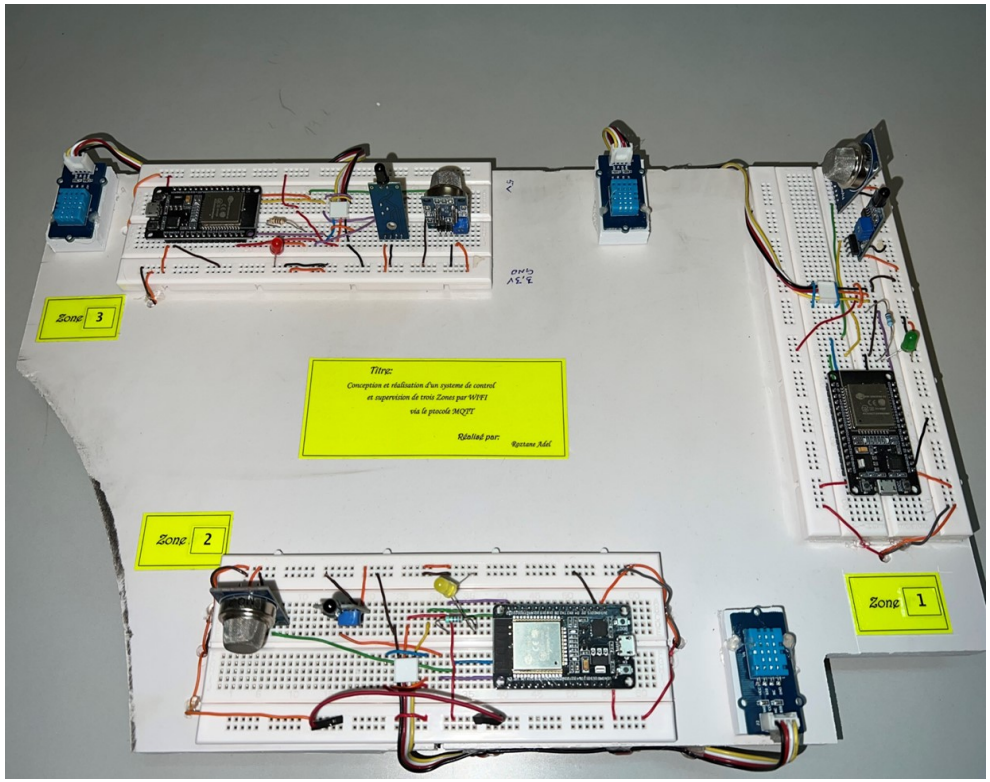


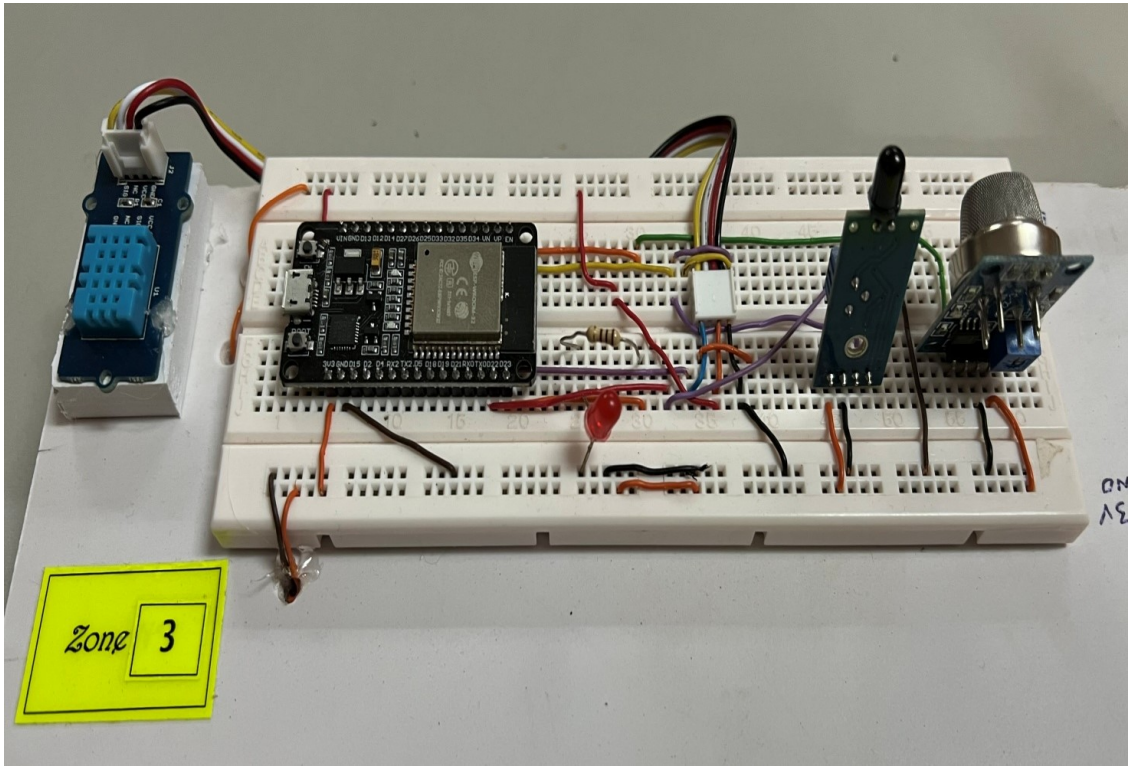


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

6. Réalisation d'un système embarqué pour la sécurité incendie





Mr. MEGNAFI Hicham (ESSA-Tlemcen)

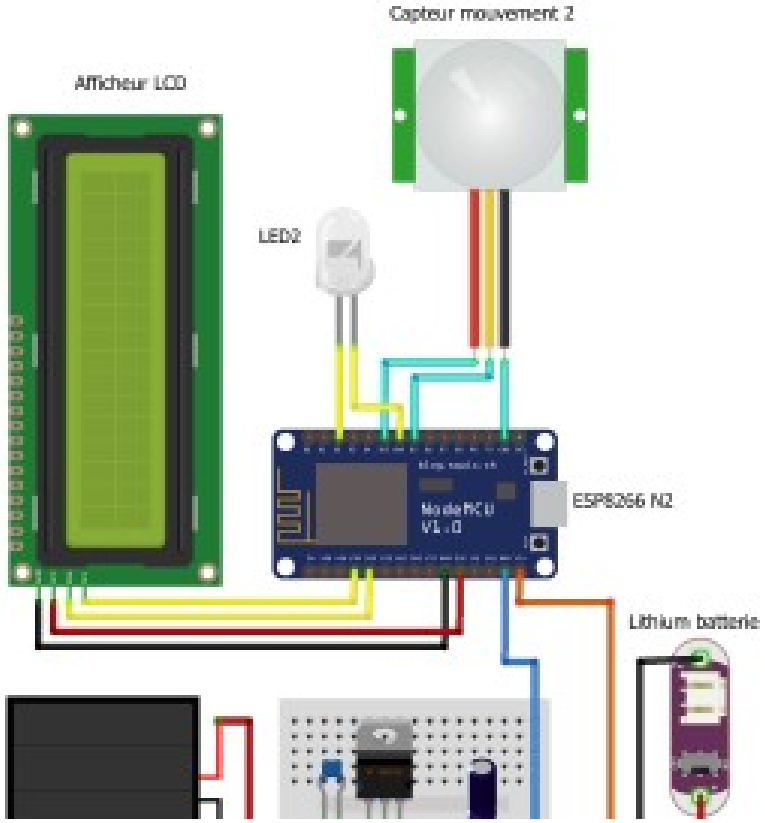
Projets réalisés

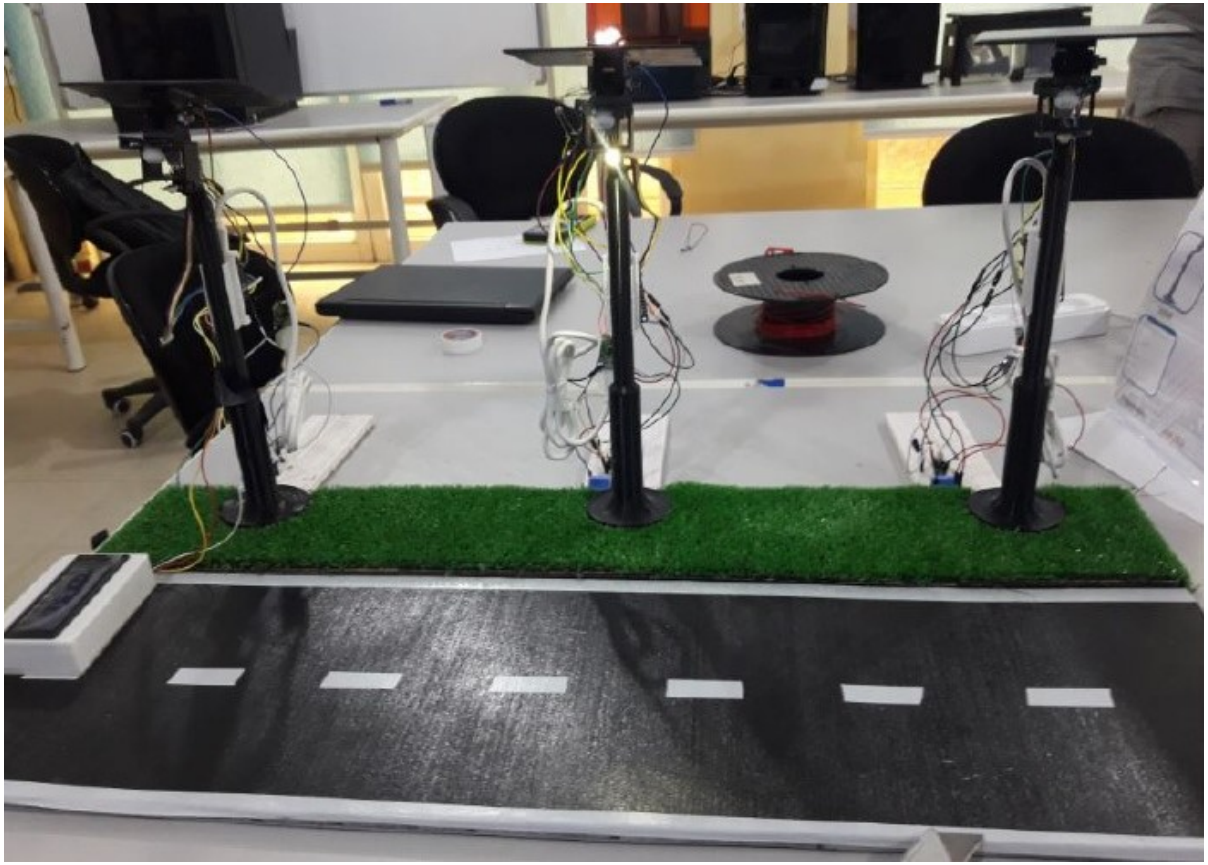
6. Réalisation d'un système embarqué pour la sécurité incendie

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Projets réalisés

7. Réalisation d'un système intelligent d'éclairage Urbain





Mini projet

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

(1/2)

1	Réalisation d'un système de supervision de rythme cardiaque des malades à distance.
2	système de surveillance de température.
3	Robot suiveur de ligne
4	Commande automatique d'une barrière de parking.
5	Conception et réalisation d'un système permettant d'afficher le nombre des personnes dans une salle.
6	Réalisation d'un bras robotique

Mini projet

-La date d'affectation des mini projets : 02/02/2023

-Possibilité de la proposition des mini projets par des étudiants.

-Chaque projet peut être fait par 4 étudiants en max

-La date de la présentation des projets : ***09/03/2023***

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Plan de Cours

1. Projets réalisés
2. ***Introduction***
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

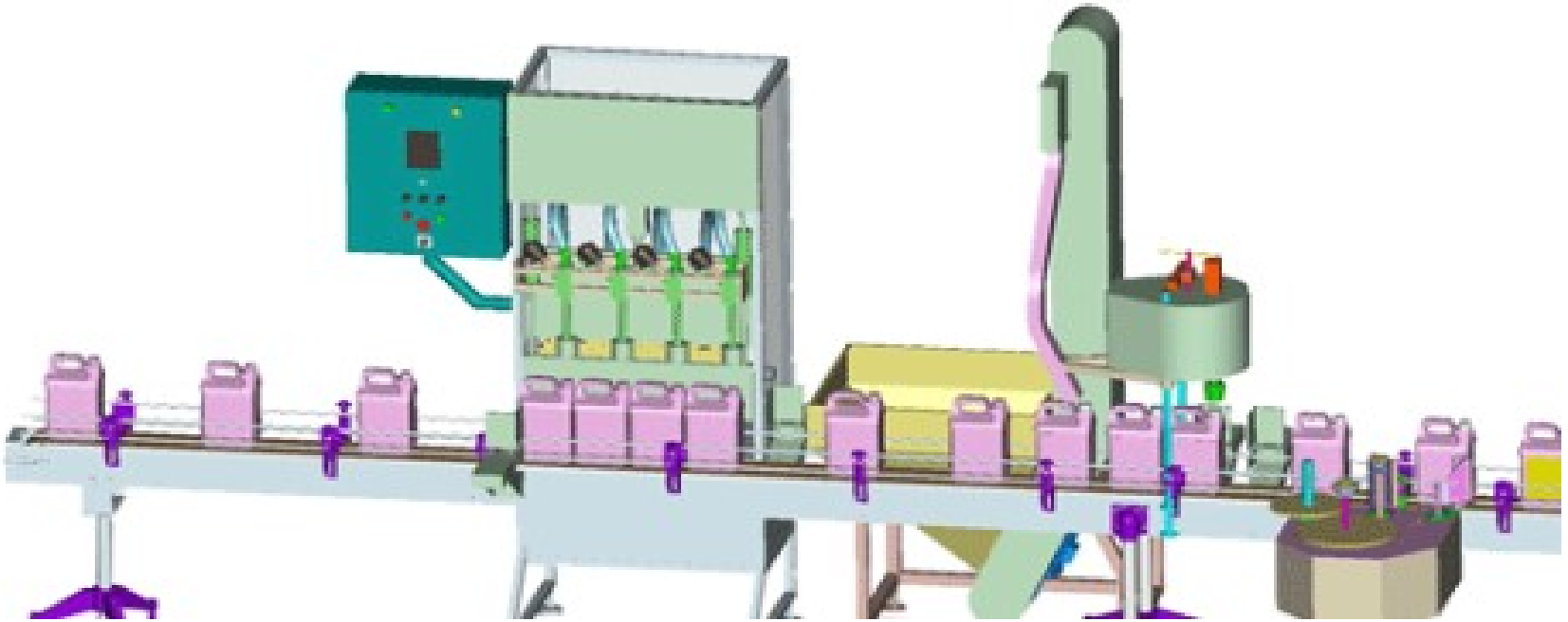
2.1 Exemple 1
- *Problématique*
- *Cahier de charge*
- *Solution 1*
- *Solution 2*
- *Inconvénients*
- *Meilleure solution*

2.2 Exemple 2
2.3 Historique
2.4 Les Microcontrôleurs
2.5 Qu'est ce qu'un microcontrôleur
2.6 L'intérêt des microcontrôleur

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Exemple 1 (1/6)

Problématique



Exemple 1 (2/6)

Cahier de charge :

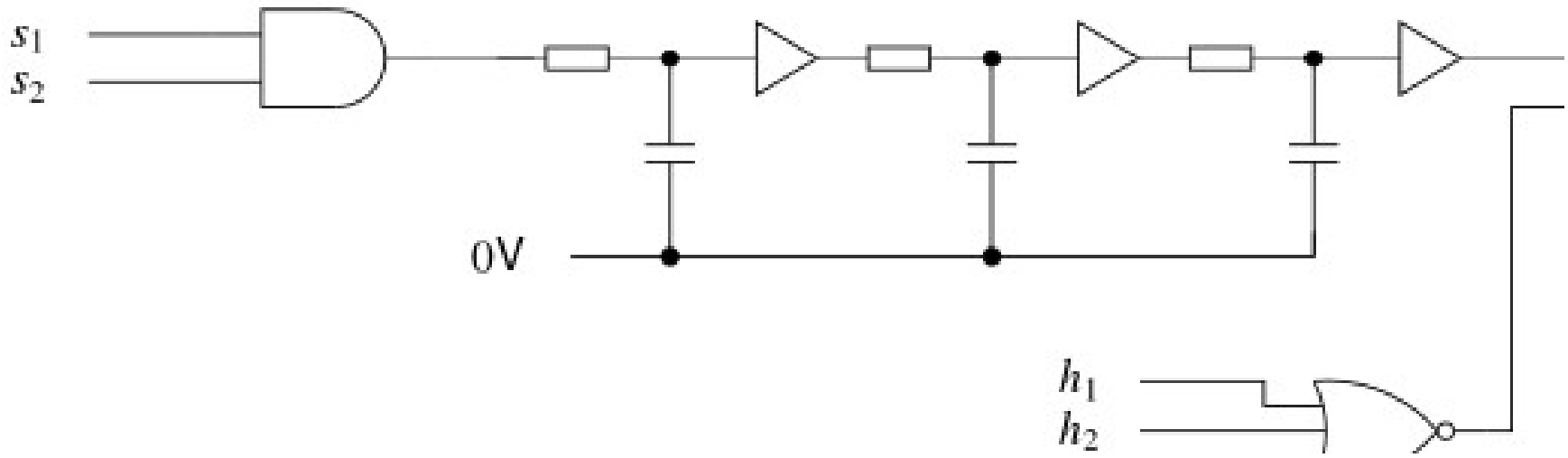
-Un objet est détecté lorsque deux capteurs le signalent.

-Trois boutons d'arret d'urgence doivent être pris en compte.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Exemple 1 (3/6)

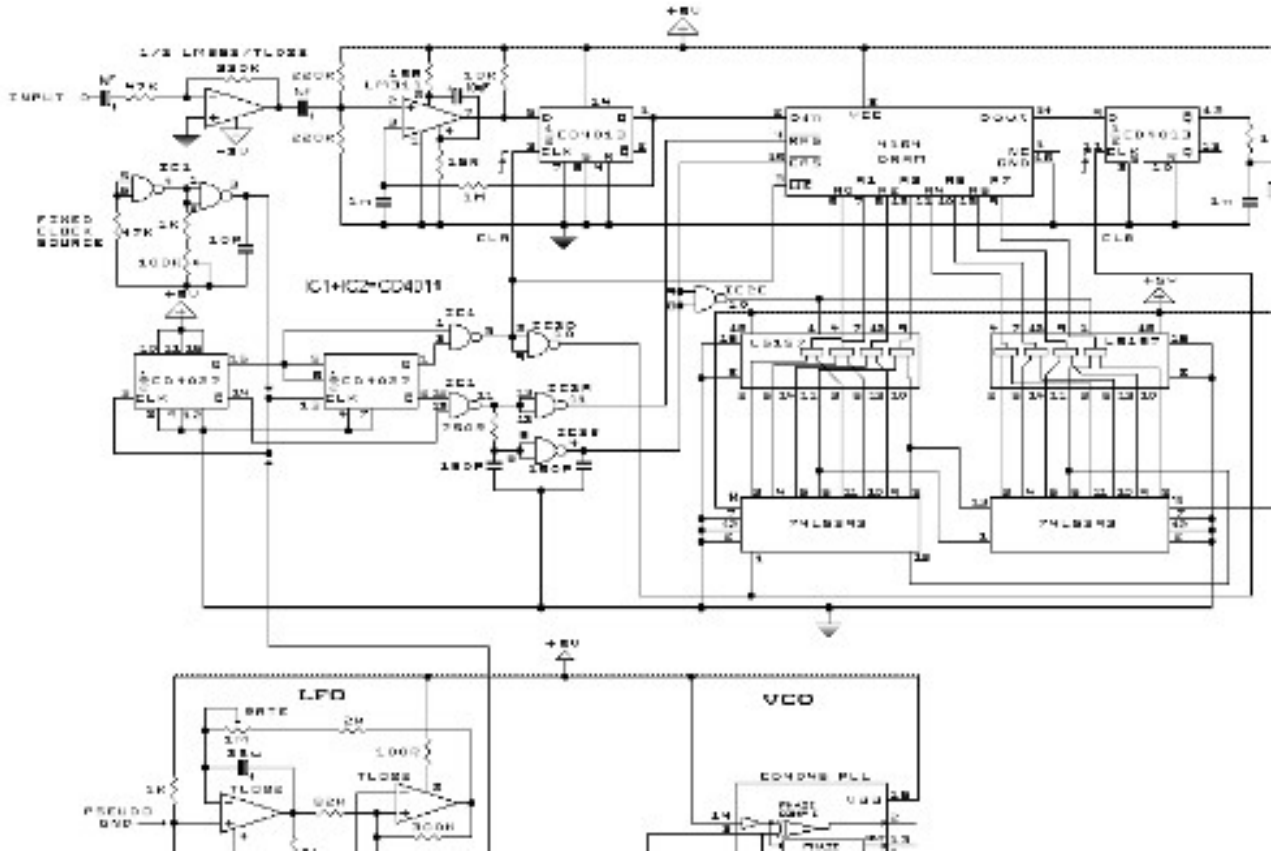
Solution 1 :



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Exemple 1 (4/6)

Solution 2 :



Exemple 1 (5/6)

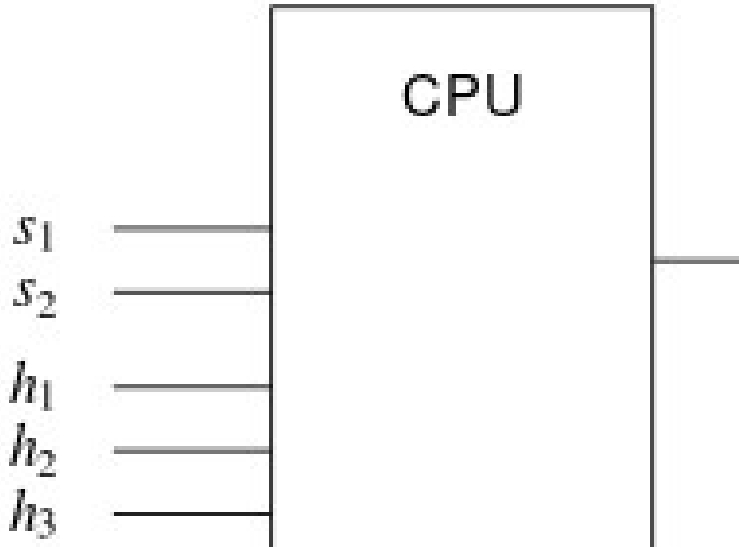
Inconvénients

- Circuits complexes.
- Difficultés de mise au point :
  - Mélange d'éléments analogiques et numériques.
  - Seuils et délais fixés par la valeur des composants.
- Manque de robustesse et de flexibilité :
  - Paramétrage difficile ou impossible.
  - Ensemble de fonctionnalités limité et peu ou pas extensible.
  - . . .
- Solutions couteuses.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Exemple 1 (6/6)

Meilleure solution

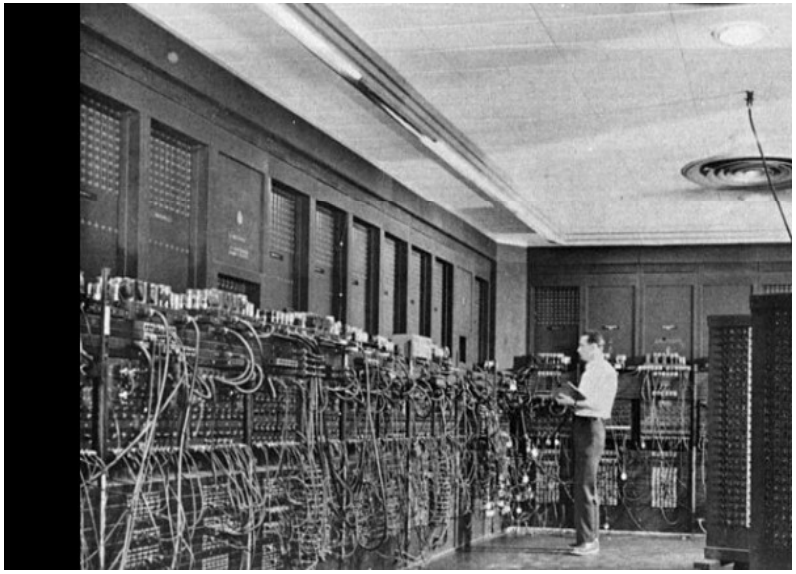


Exemple 1 (5/6)

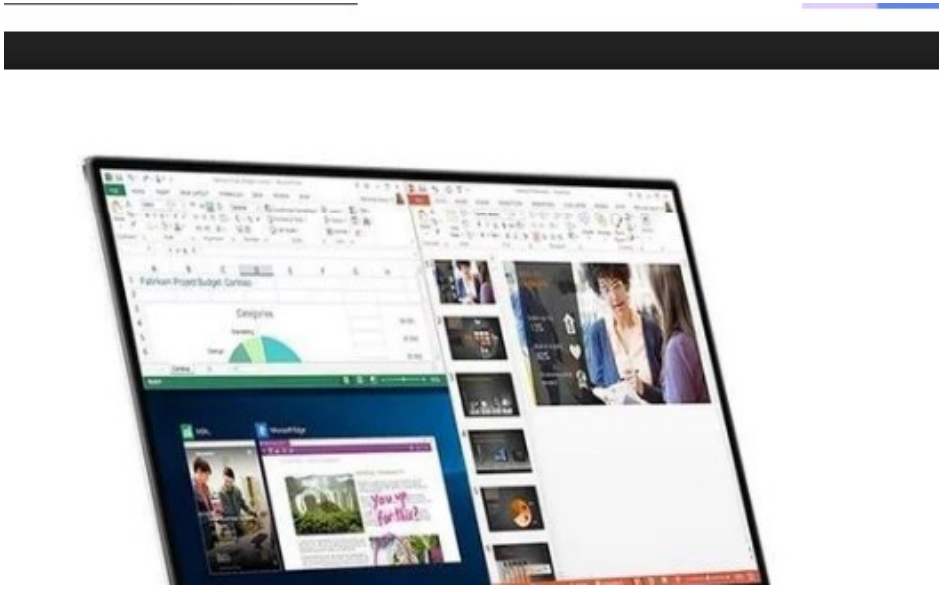
Avantages

- Circuit simple, facile à concevoir et à construire
- Difficultés déplacées du matériel vers le logiciel
- Fonctionnalités facilement extensibles :
  - communications,
  - temporisations,
  - paramétrage,
  - gestion d'un historique,
  - alarmes,
  - ...
- Solution compacte et bon marché.

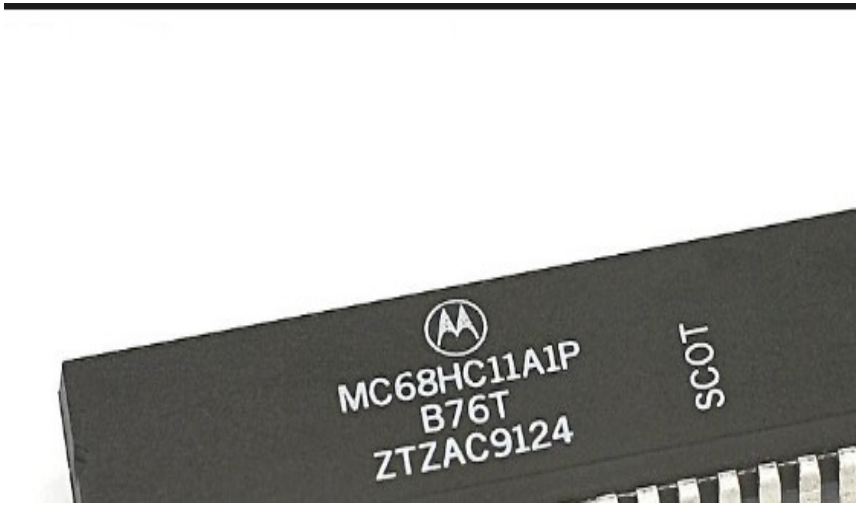
Ordinateur 1964



Ordinateur en 2019



Il existe aussi les Microcontrôleurs

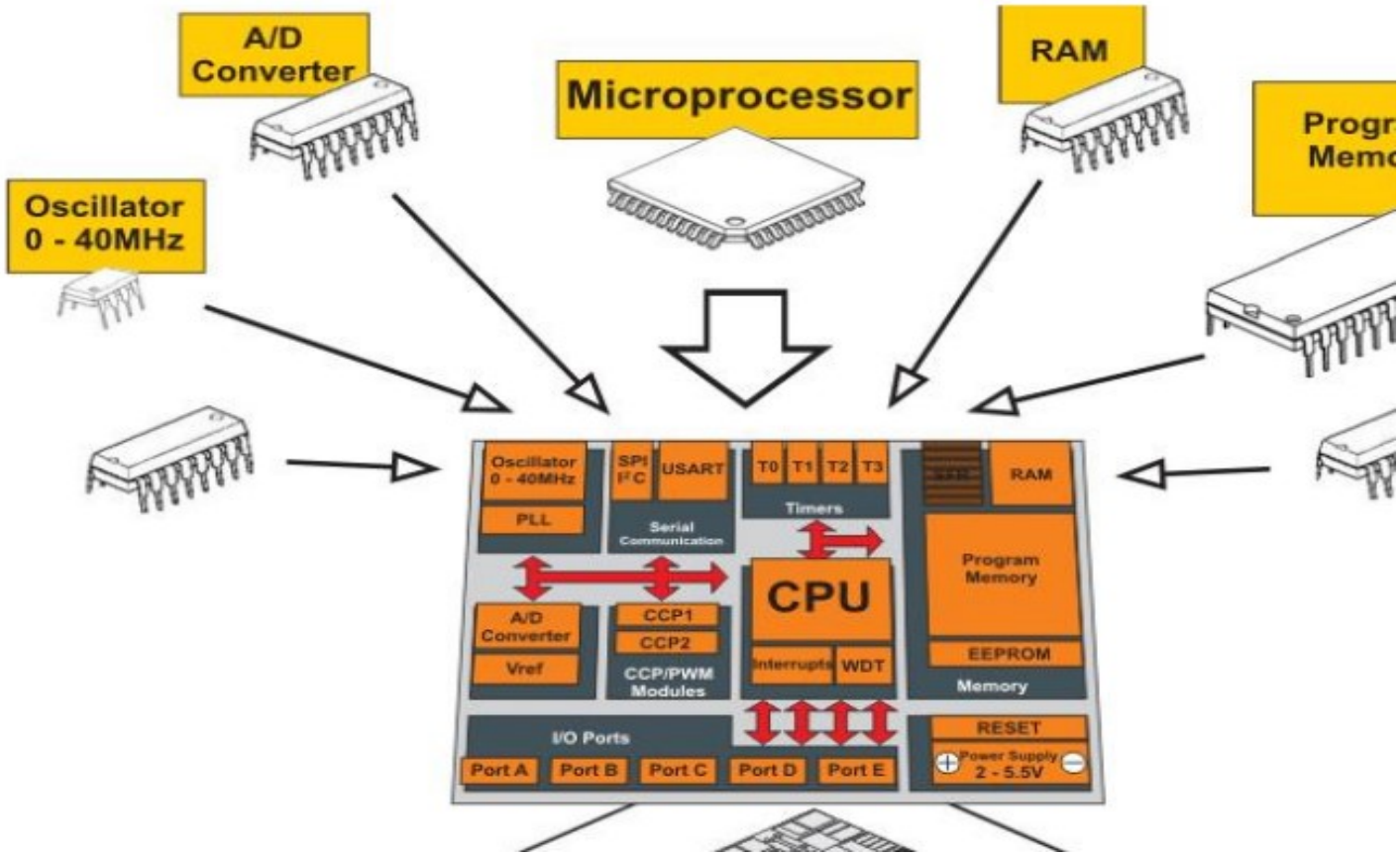


Les Microcontrôleurs

- Un microprocesseur (MP) est une unité central de traitement (CPU) Central Processing Unit, Il représente l'unité de traitement d'un micro-ordinateur,
- Un ordinateur est une machine qui permet le traitement de données selon une suite d'instructions appelées programme,
- Un ordinateur est une combinaison de matériels et logiciels (hardware and software) qui traitent les données selon un algorithme écris par un programmeur (user software),
- un ordinateur = CPU + mémoire + I/O

Définition : Un microcontrôleur est un micro-ordinateur conçus pour le contrôle industriel,

Les Microcontrôleurs



Qu'est ce qu'un microcontrôleur (µc)

Un microcontrôleur (en notation abrégée c, ou uc ou encore MCU en anglais) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur :

- Processeur,
- mémoires (mémoire morte pour le programme, mémoire vive pour les données),
- unités périphériques et interfaces d'entrées-sorties.

Qu'est ce qu'un microcontrôleur (µc)

Les microcontrôleurs se caractérisent par :

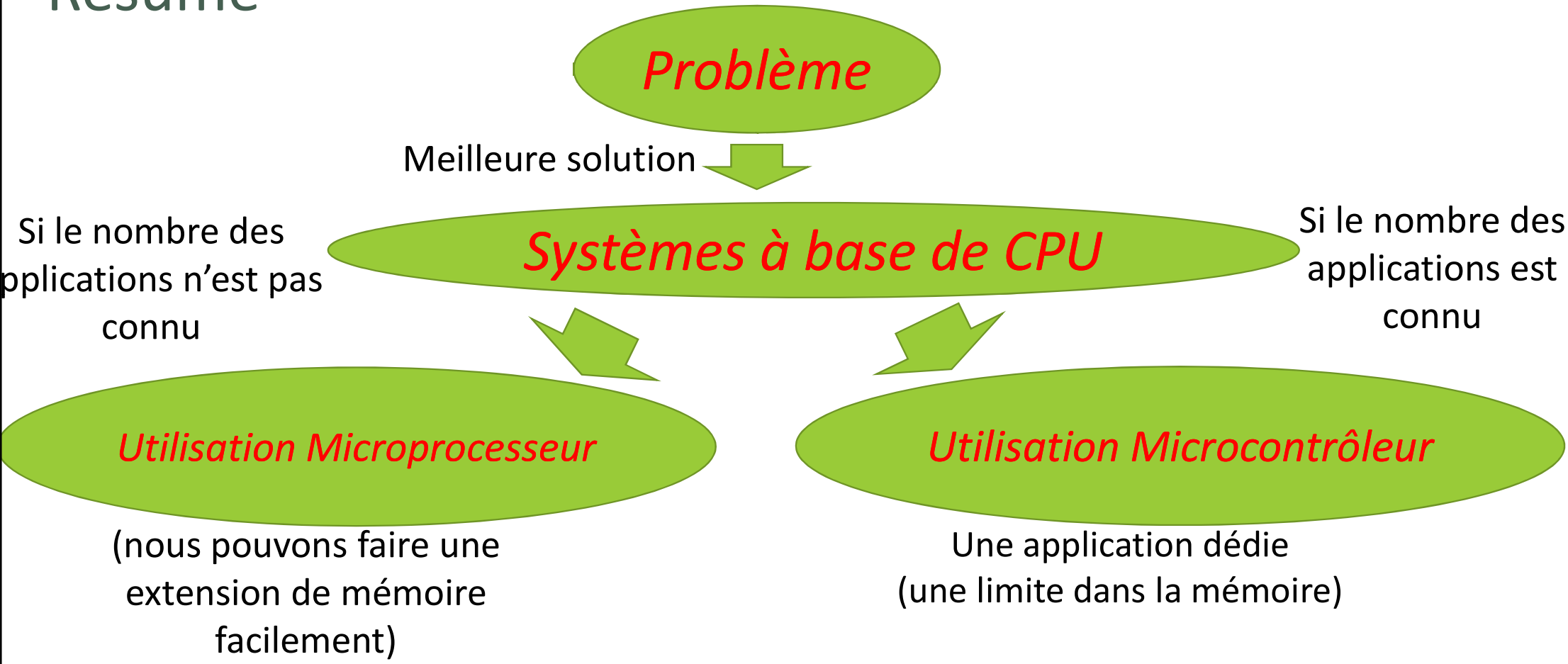
- un plus haut degré d'intégration,
- une plus faible consommation électrique,
- une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) ,
- un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

L'intérêt des microcontrôleur

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc

Résume


Problème
Meilleure solution
Si le nombre des applications n'est pas connu
Systèmes à base de CPU
Si le nombre des applications est connu
Utilisation Microprocesseur
Utilisation Microcontrôleur
(nous pouvons faire une extension de mémoire facilement)
Une application dédie
(une limite dans la mémoire)

Plan de Cours

1. Projets réalisés
2. Introduction
3. ***Types d'architecture***
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

3.1 Types d'architecture
3.2 Architecture de Von Neumann
3.3 Exécution d'instructions Vonn Neumann
3.4 Architecture de Harvard
3.5 Exécution d'instructions Harvard
3.6 Comparaison Von Neumann Vs Harvard

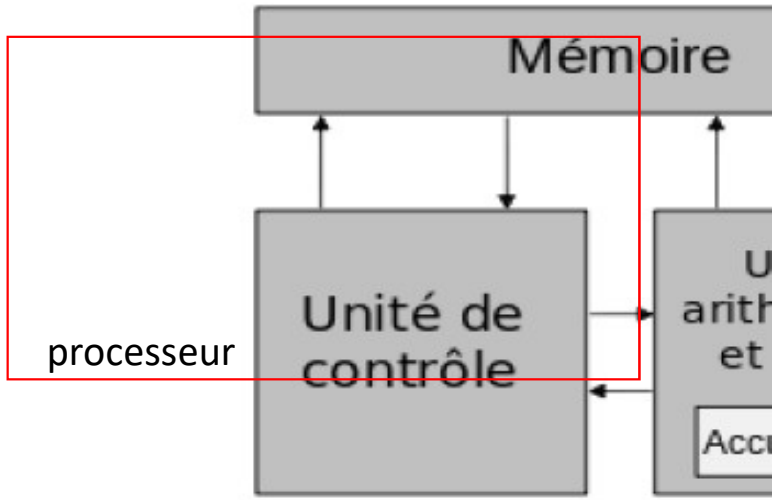
Types d'architecture

Les éléments d'une architecture :

- CPU,
- Mémoire,
- Péréphiriques I/O,

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Architecture de Von Neumann



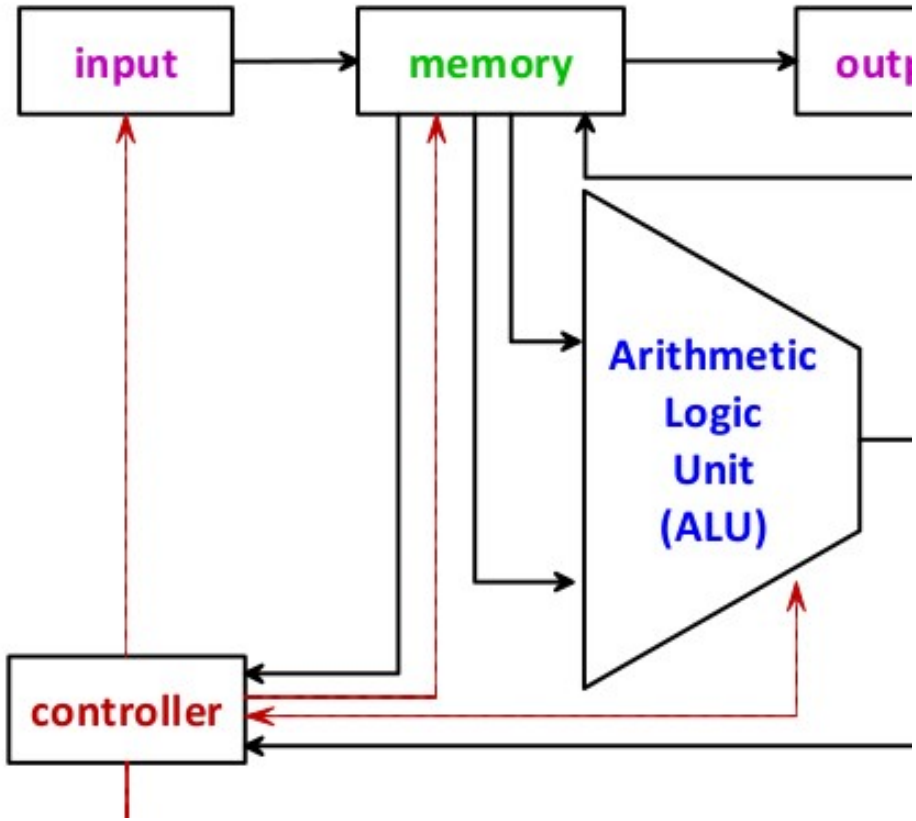
Structure de VON NEUMANN : La première structure a être établie.

Architecture de Von Neumann

- Traitement numérique au niveau de l'unité arithmétique est logique (ALU) ,
- Programmable à travers un ensemble d'instructions,
- mémoire de données intégrée,
- Mémoire de programmes intégrées,
- Input/Output automatique
- Séquencement automatique des instructions a exécutées par le contrôleur,
- Un seul chemin d'accès à la mémoire
  - Un bus de donnée (programme et donnée)
  - Un bus adresses (programme et donnée)

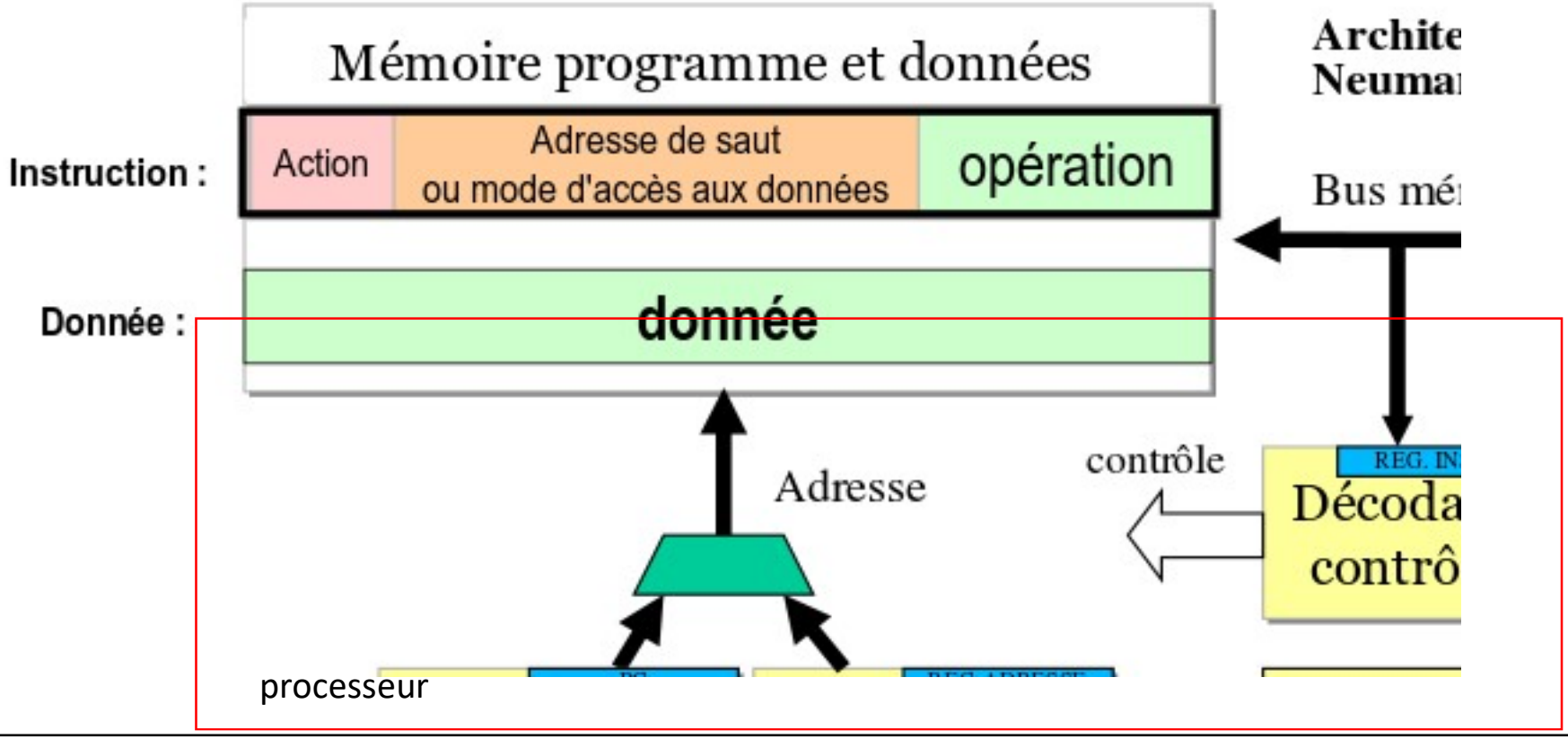
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Architecture de Von Neumann

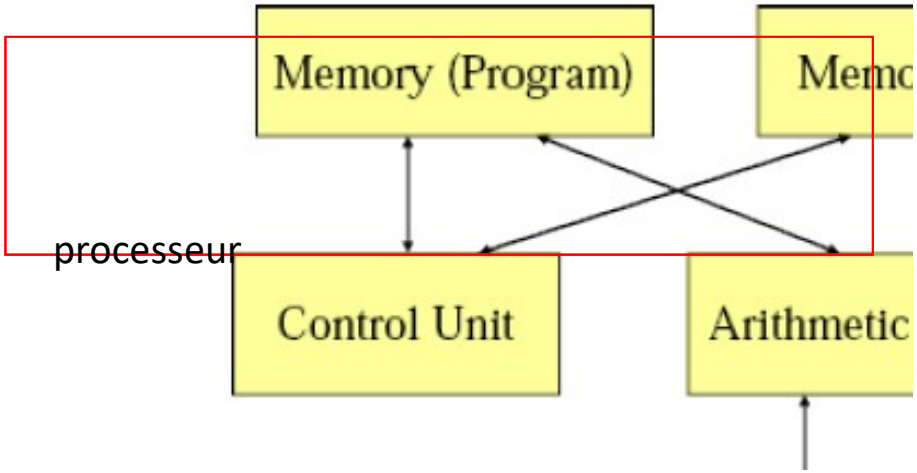


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Exécution d'instructions Vonn Neumann



Architecture de Harvard



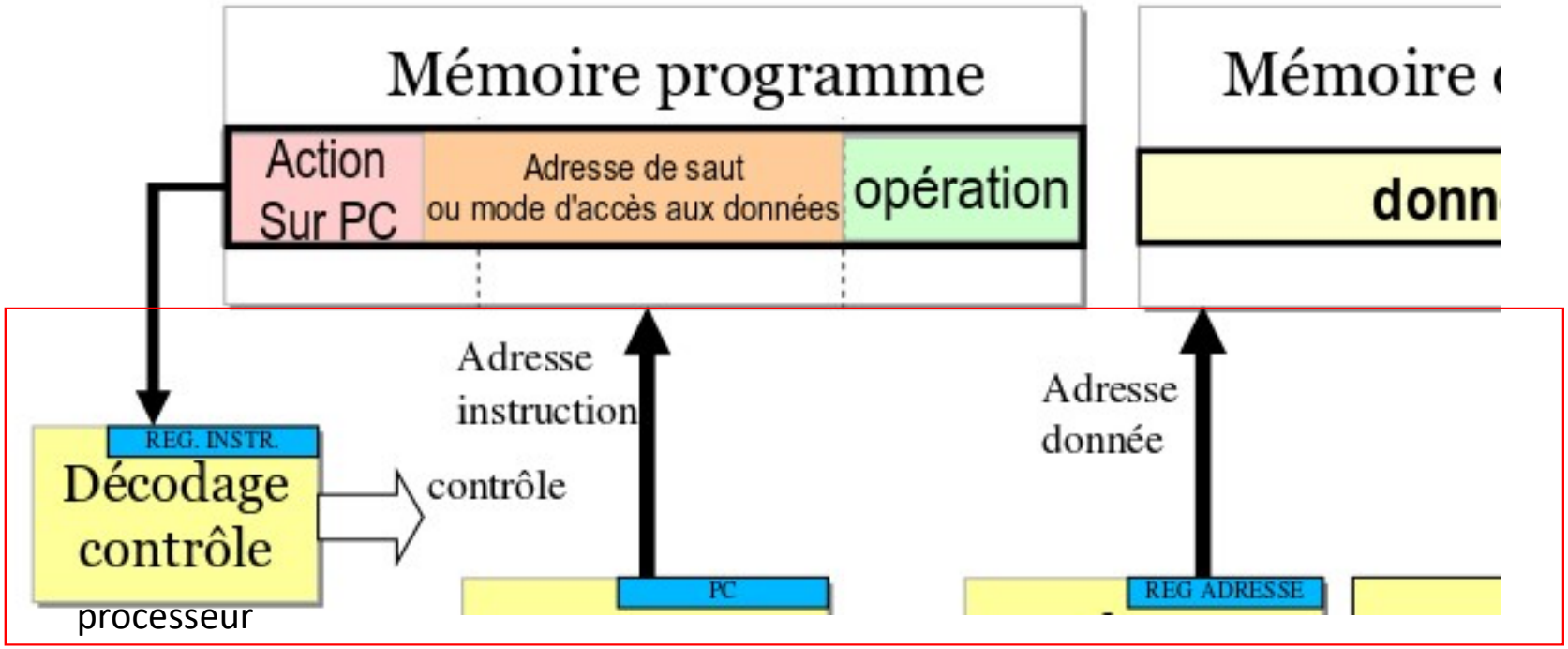
Structure d'HARVARD : conception qui sépare physiquement la mémoire en deux parties. Mémoire des données et Mémoire des programmes.

Architecture de Harvard

- Séparation des mémoires programme et données
  - Un bus de données programme,
  - Un bus de données pour les données,
  - Un bus d'adresse programme,
  - Un bus d'adresse pour les données.
- Meilleure utilisation du CPU :
- Chargement du programme et des données en parallèle

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

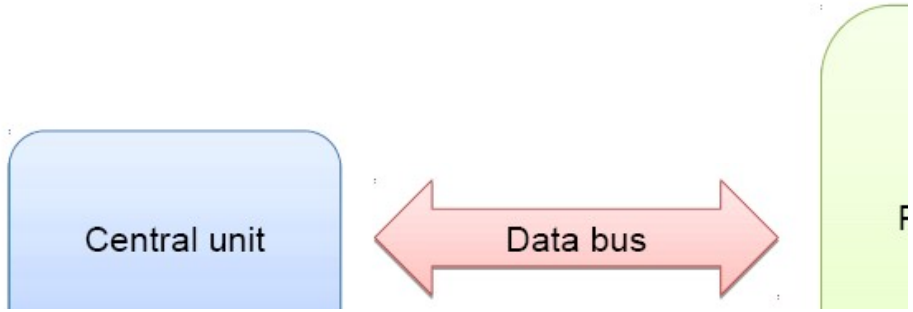
Exécution d'instructions Harvard



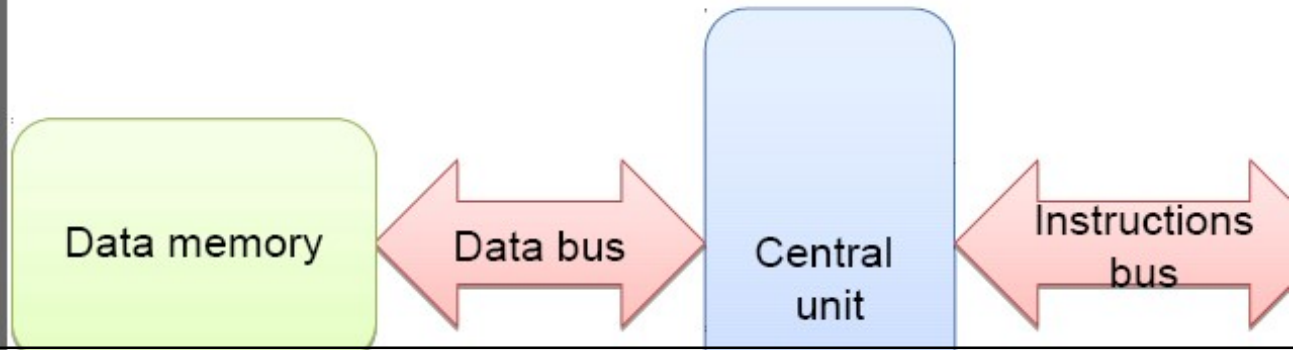
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Comparaison Von Neumann Vs Harvard

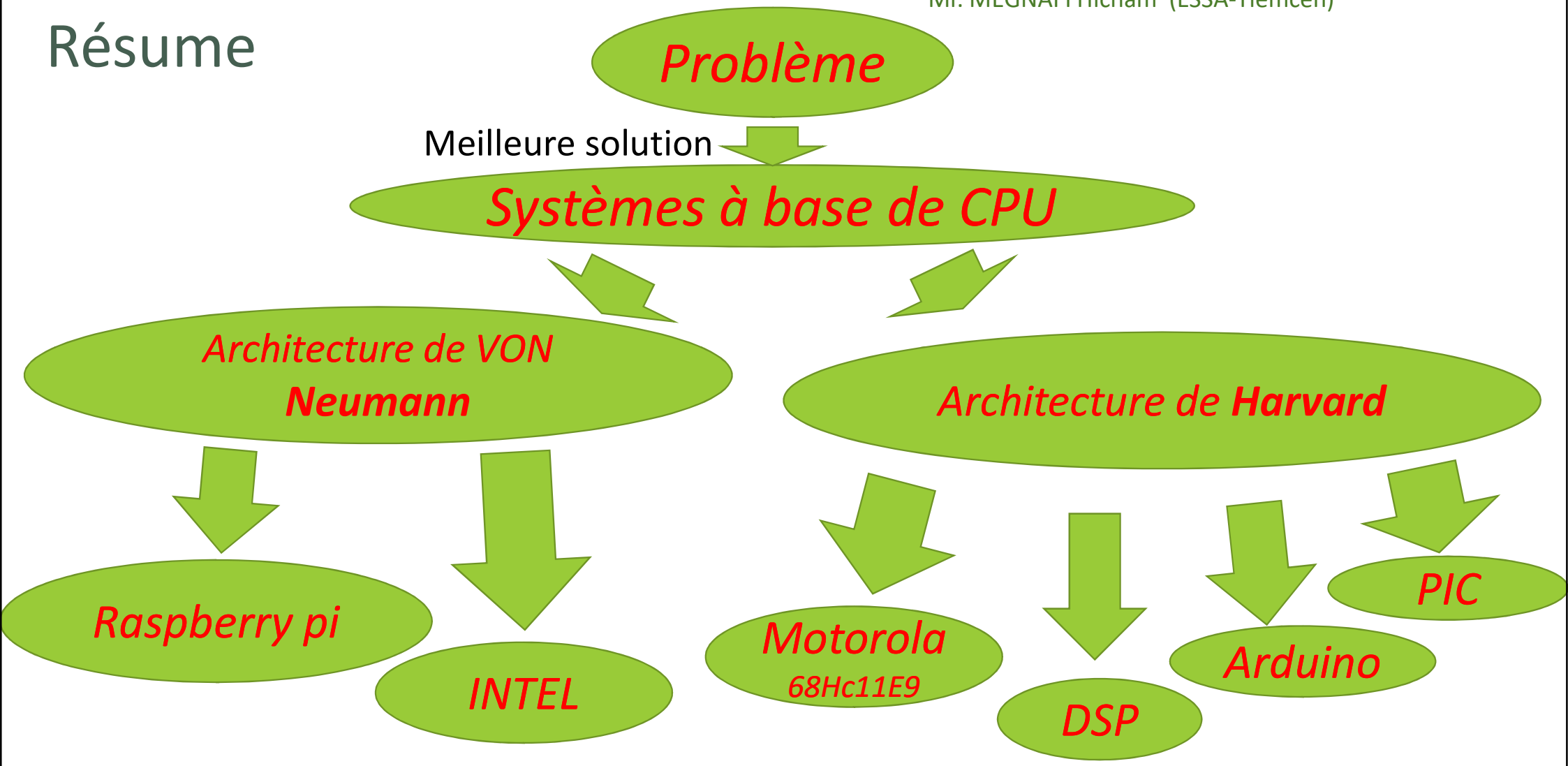
Architecture de Von Neumann



Architecture de Harvard



Résume


Problème
Meilleure solution
Systèmes à base de CPU
Architecture de VON Neumann
Architecture de Harvard
Raspberry pi
INTEL
Motorola 68Hc11E9
DSP
Arduino
PIC

Architecture de VON
Neumann

Raspberry pi

INTEL

Architecture de **Harvard**

- **Motorola** 68Hc11E9
- **DSP**
- **Arduino**
- **PIC**

Plan de Cours

1. Projets réalisés
2. Introduction
3. Types d'architecture
4. **Les Processeurs**
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

4.1 Qu'est-ce qu'un processeur ?

4.2 Architecture processeur

- ***Principe de fonctionnement de l'UAL et l'UC***
- ***Unité de contrôle***
  - ***Horloge***
  - ***Séquenceur***
- ***Unités de calcul***
- ***Registre***

4.3 jeux d'instructions

4.4 Types de jeux d'instructions :

- ***CISC***
- ***RISC***

4.5 Utilisation de pipeline

Qu'est-ce qu'un processeur ?

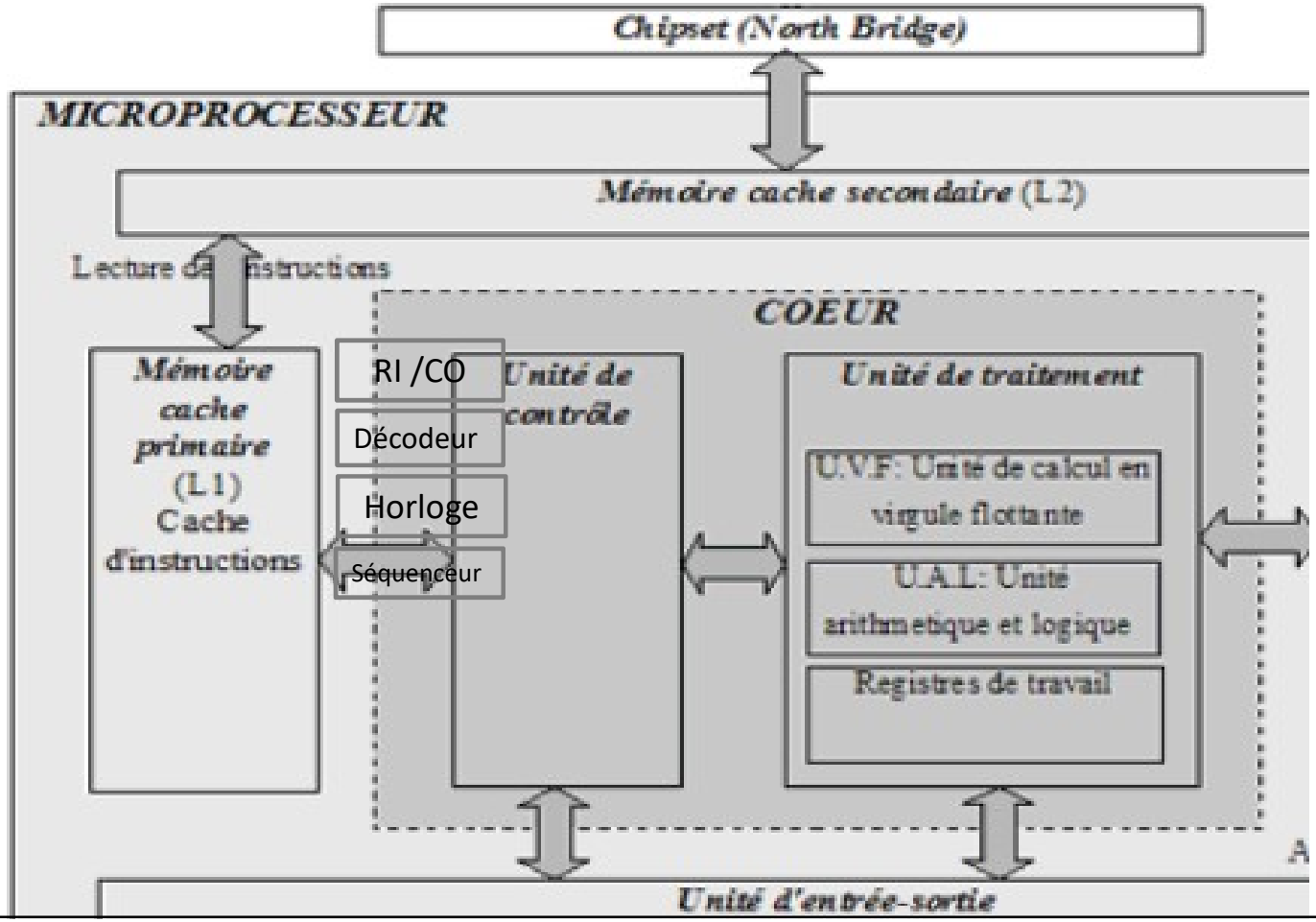
Définition : Le processeur est un composant qui exécute des instructions séquentiellement (programme) à partir de données,

Il possède (généralement) :

- **Unité de calcul (UAL) :** Arithmétique,Logique
- **Unité de commande :** Coordination générale , Lecture et décodage des instructions
- **Registres**
- **Horloge**
- **Mémoire cache**

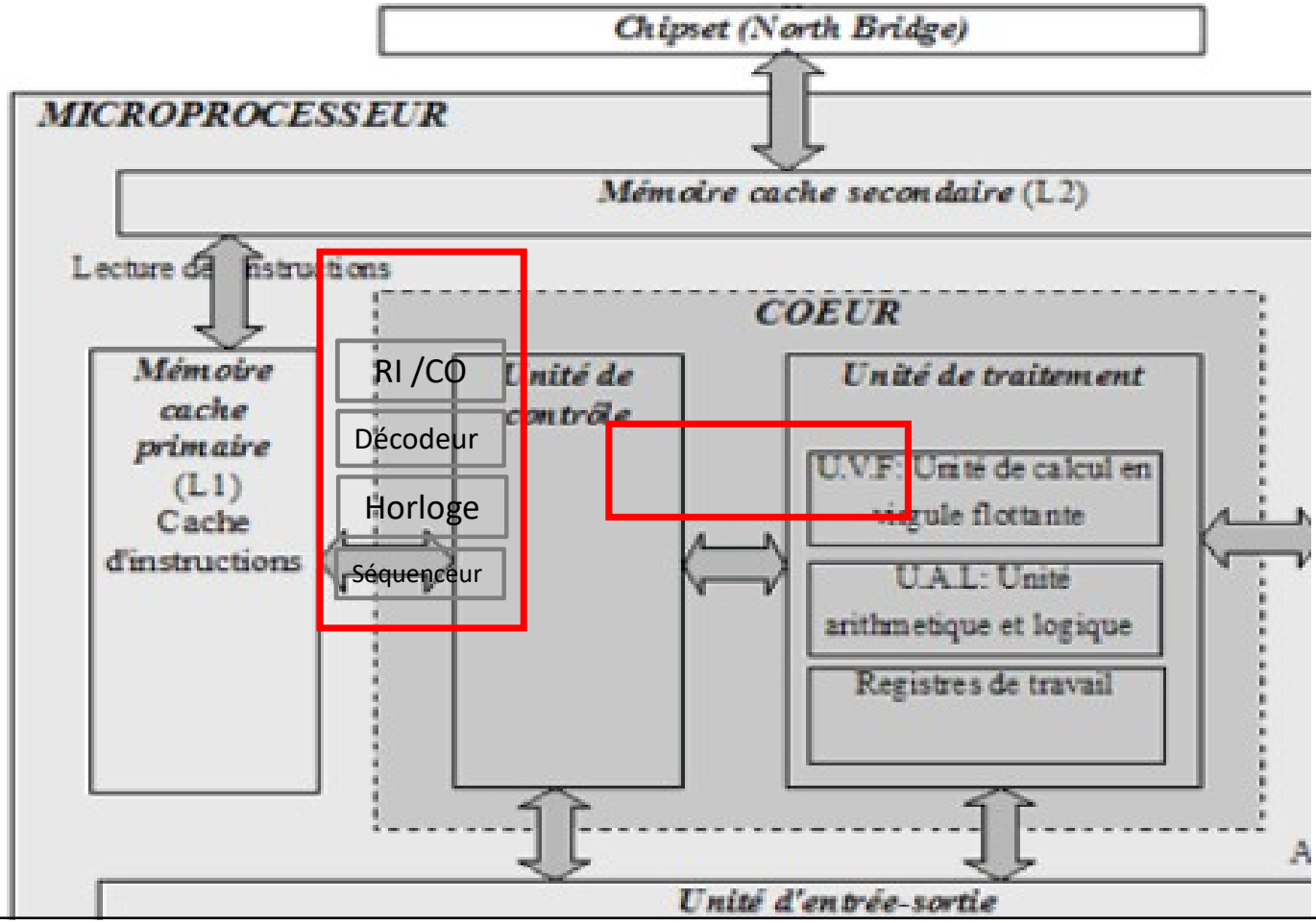
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Architecture Processeur



Mr. MEGNAFI Hicham (ESSA -Tlemcen)

Architecture Processeur

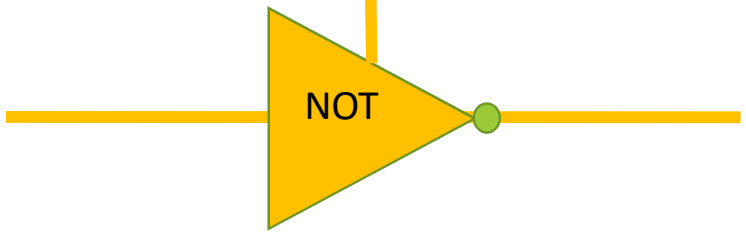


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Principe de l'Unité Arithmétique et logique UAL l'Unité de Contrôle UC

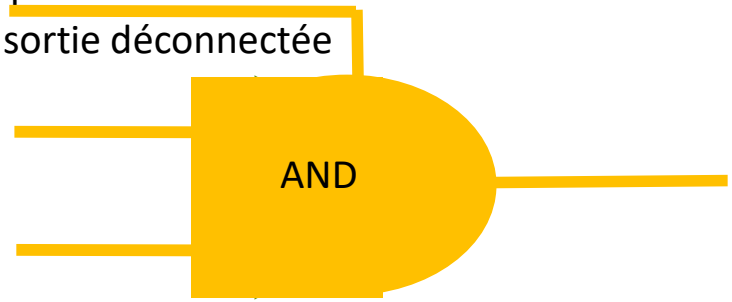
Sélection 1 :porte active

0 sortie déconnectée



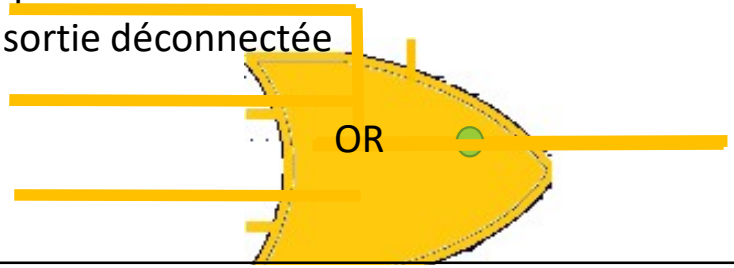
Sélection 1 : porte active

0 sortie déconnectée



Sélection 1 : porte active

0 sortie déconnectée

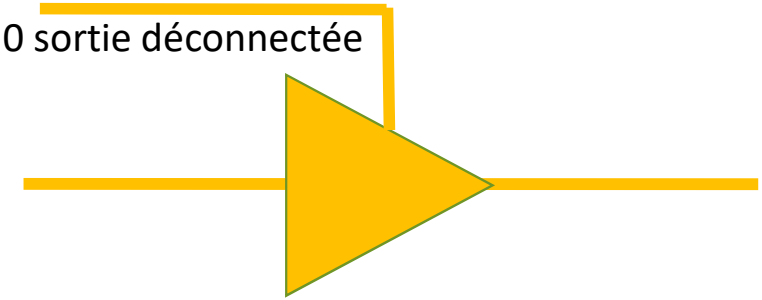


Trois portes logiques avec les quelles il est possible de résoudre n'importe quel problème en relation avec la technologie.

L'ère du numérique à trois états
0 1 et HZ (haute impédance)

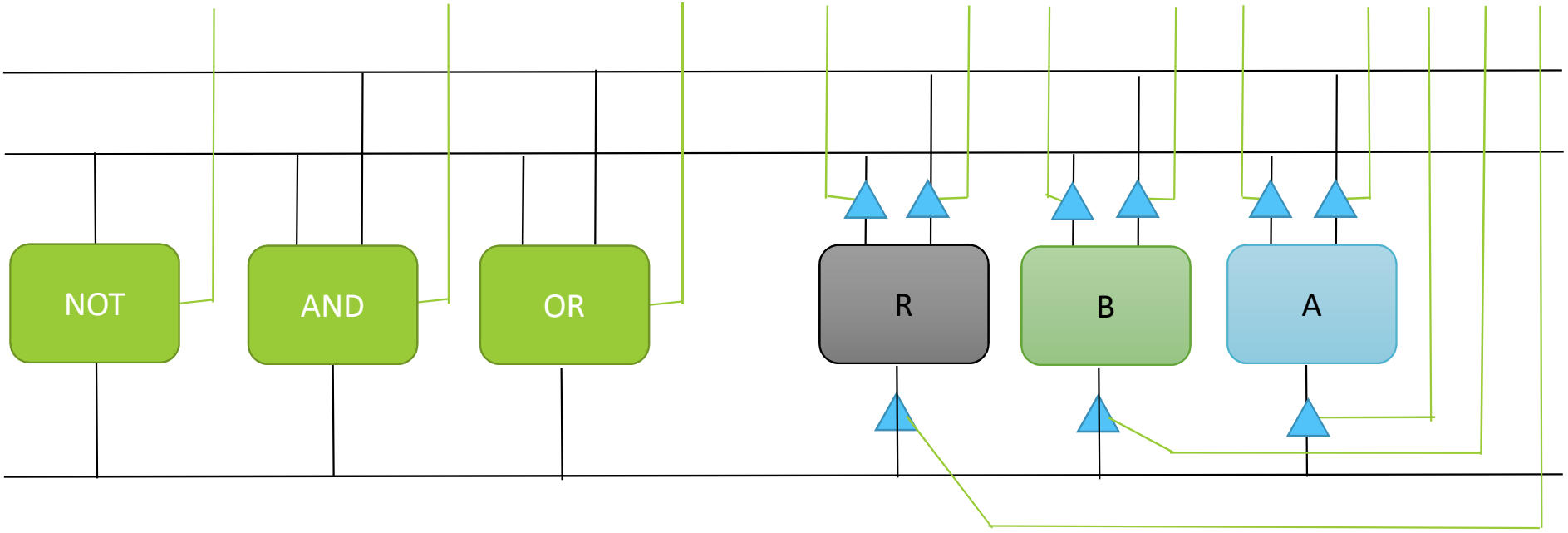
Sélection 1 : sortie = entrée

0 sortie déconnectée



Principe de fonctionnement de l'UAL et l'UC

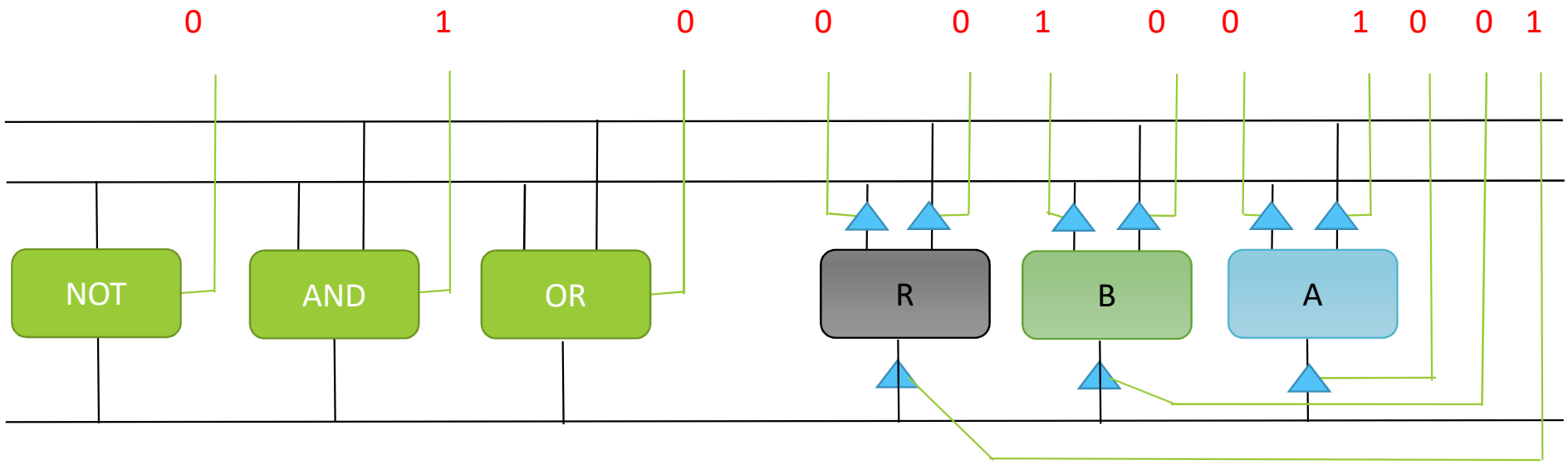
R=(Not(A) And Not(B)) Or (A And B)



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

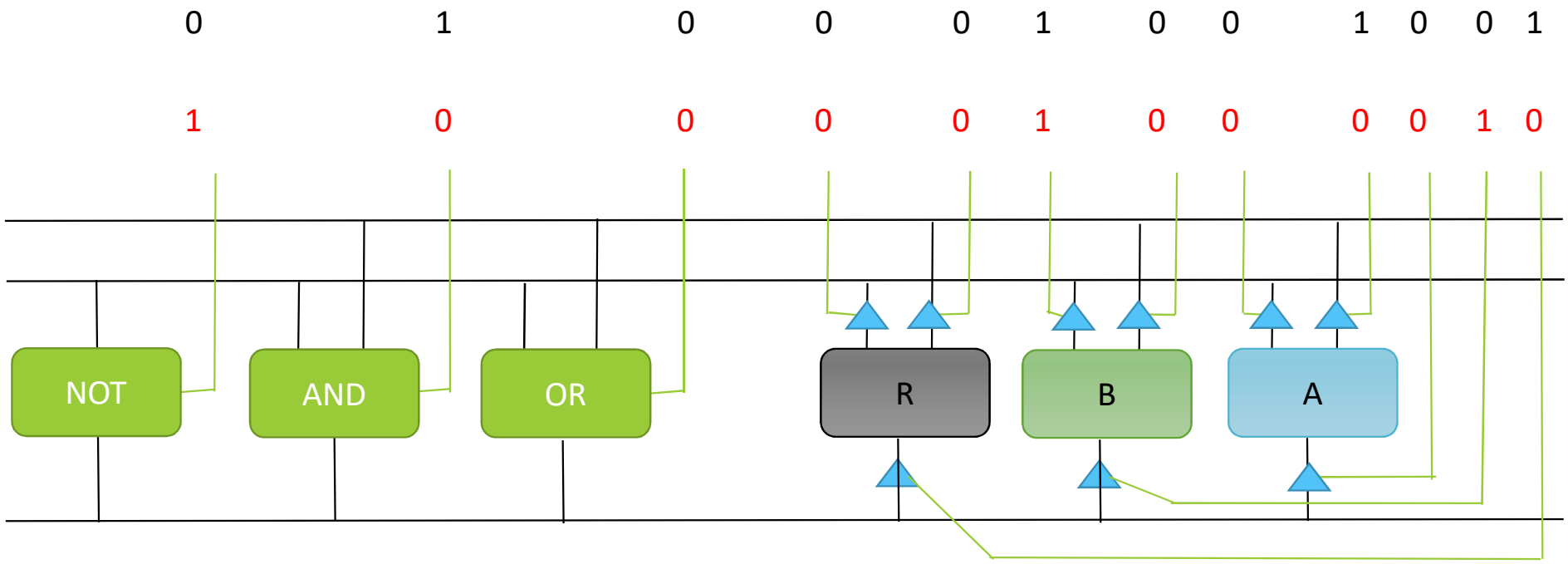
Principe de fonctionnement de l'UAL et l'UC

R=Not(A) And Not(B) Or A And B



Principe de fonctionnement de l'UAL et l'UC

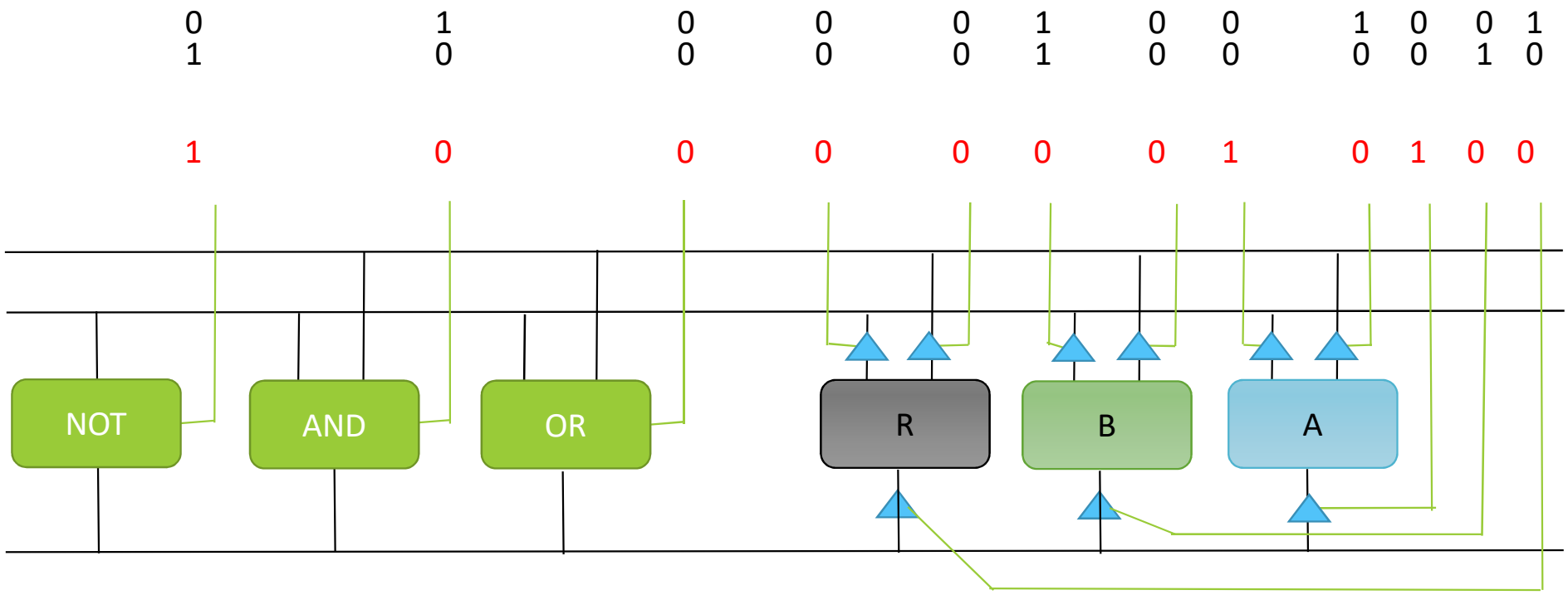
R=Not(A) And Not(B) Or A And B



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Principe de fonctionnement de l'UAL et l'UC

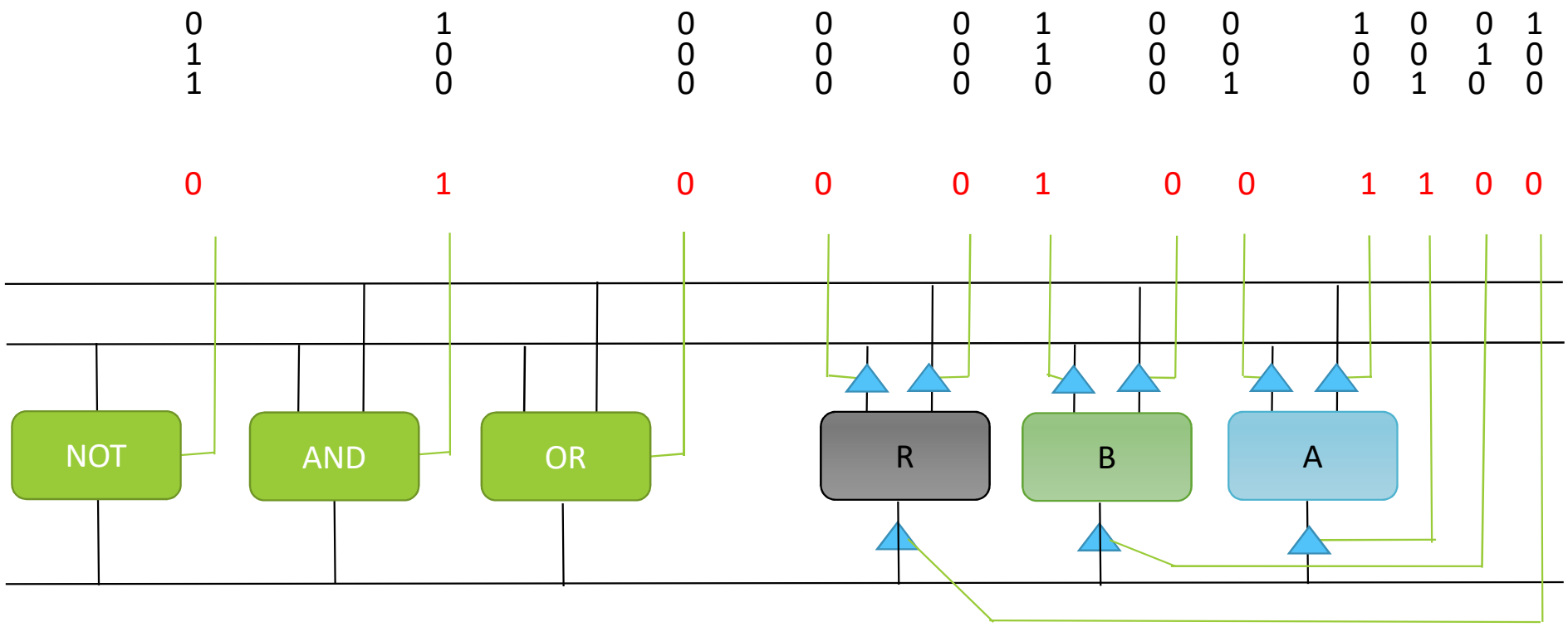
R=Not(A) And Not(B) Or A And B



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Principe de fonctionnement de l'UAL et l'UC

R=Not(A) And Not(B) Or A And B



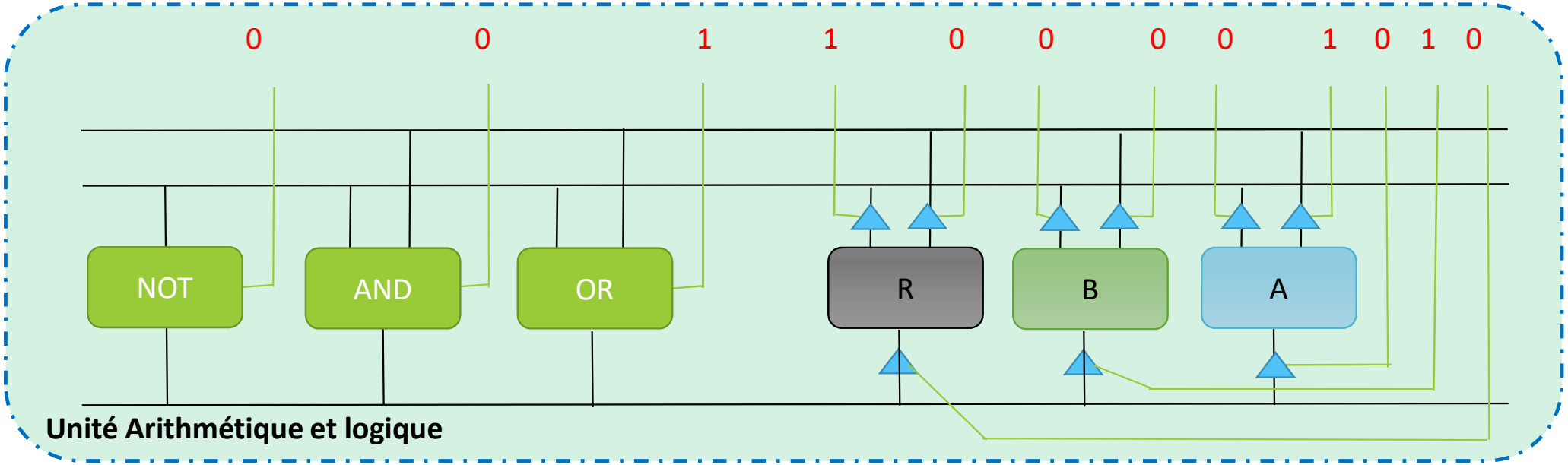
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

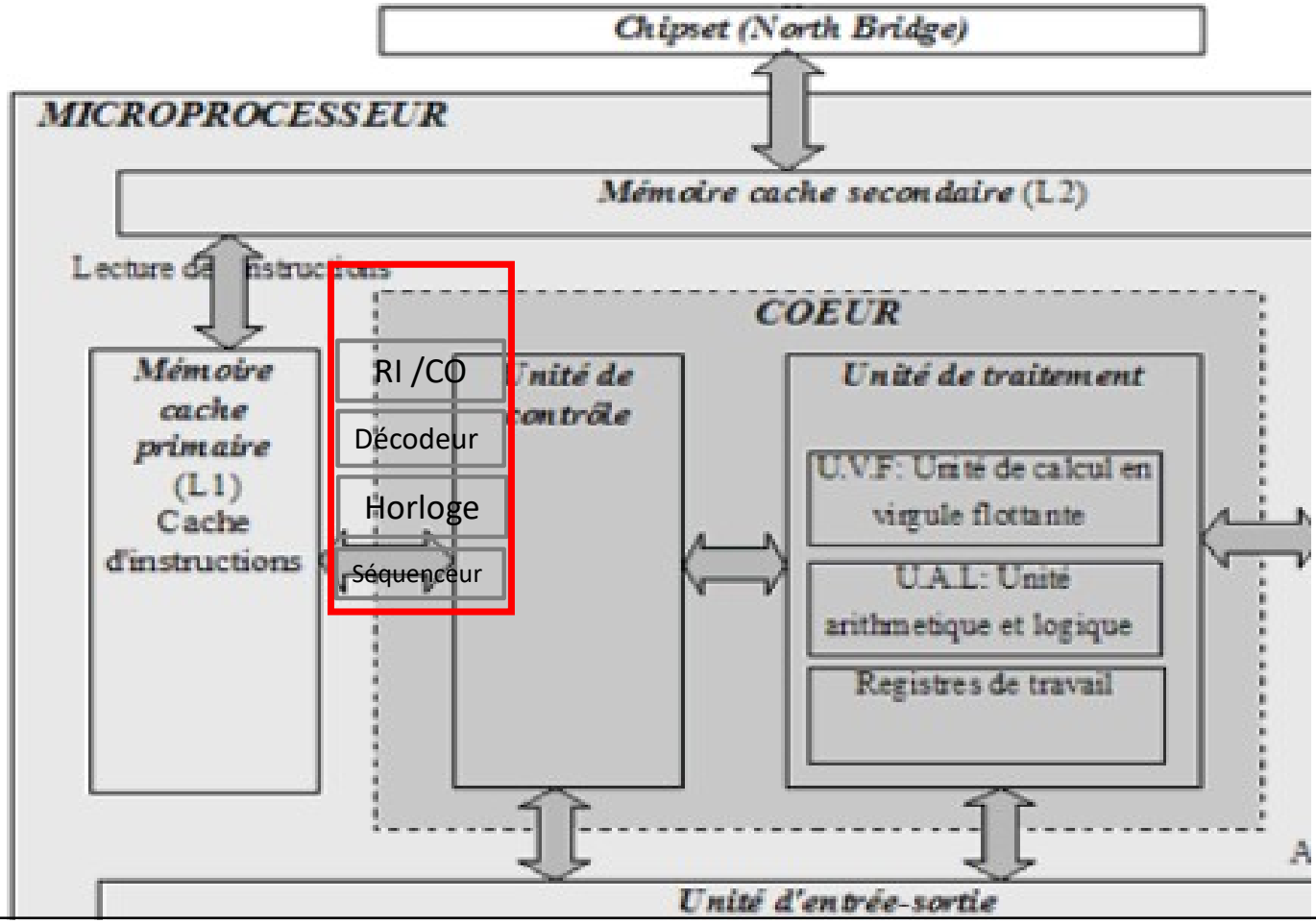
Principe de fonctionnement de l'UAL et l'UC

R = Not(A) And Not(B) Or A And B

Micro instructions : Unité de contrôle

0	1	0	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	0	0	1	0
1	0	0	0	0	1	0	1	0	1	0	0
0	1	0	0	0	1	0	0	1	1	0	0




Chipset (North Bridge)
MICROPROCESSEUR
Mémoire cache secondaire (L2)
COEUR
Mémoire cache primaire (L1) Cache d'instructions
RI /CO
Décodeur
Horloge
Séquenceur
Unité de contrôle
Unité de traitement
U.V.F: Unité de calcul en virgule flottante
U.A.L: Unité arithmetique et logique
Registres de travail
Unité d'entrée-sortie

Unité de contrôle

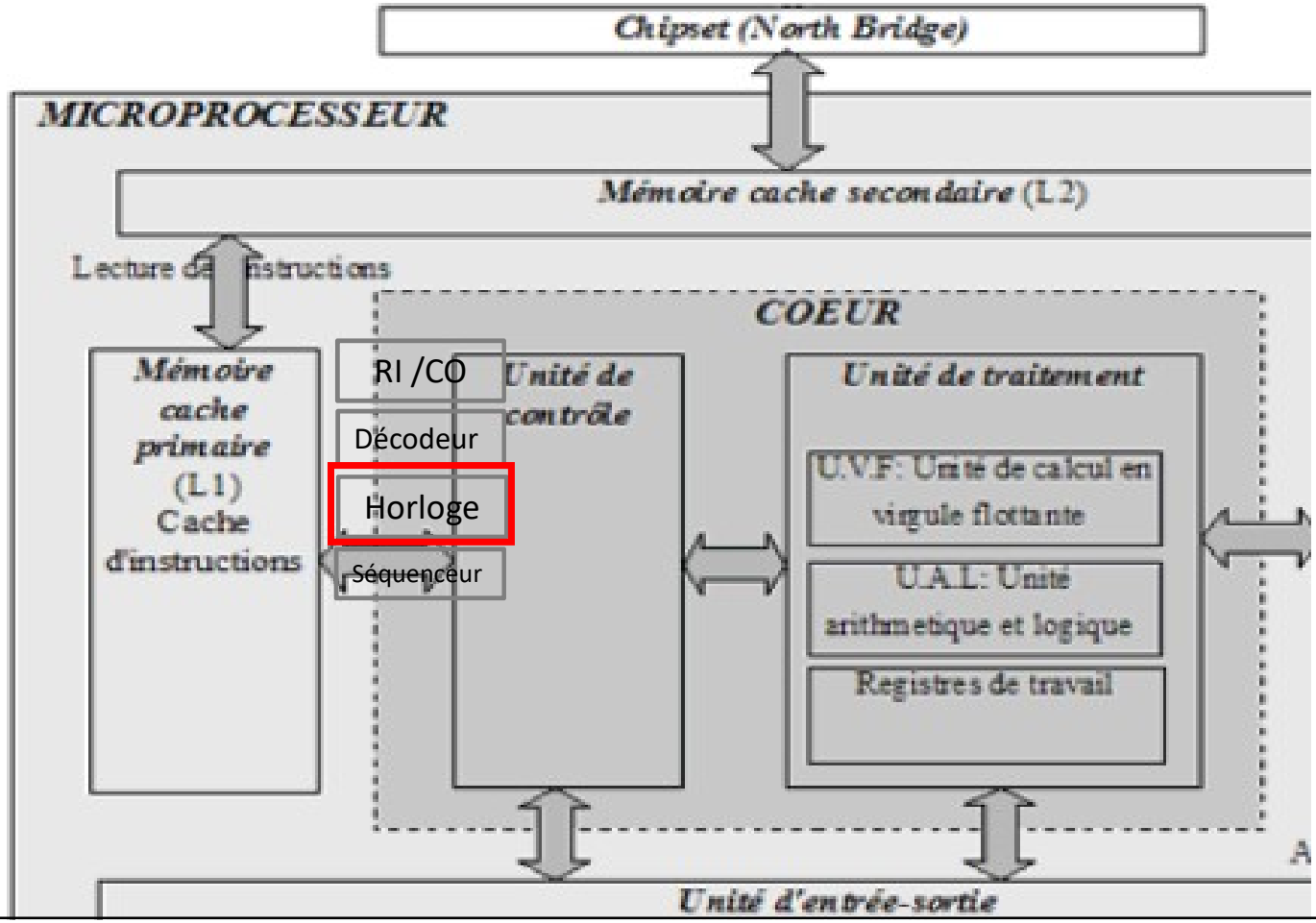
Définition

Unité qui coordonne le fonctionnement des autres éléments dans le but d'exécuter une séquence d'instructions (programme)

Constitue de plusieurs éléments :

- **Registre d'Instruction (RI)** : reçoit le ***code*** de la ***prochaine*** instruction à exécuter
- **Compteur Ordinal (CO) :** registre contenant **l'*adresse*** du mot mémoire stockant le code de la ***prochaine*** instruction
- **Décodeur** : à partir du code de l'instruction, détermine l'opération à exécuter
- **Horloge** : pour synchroniser les éléments
- **Séquenceur** : coordonne le tout

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

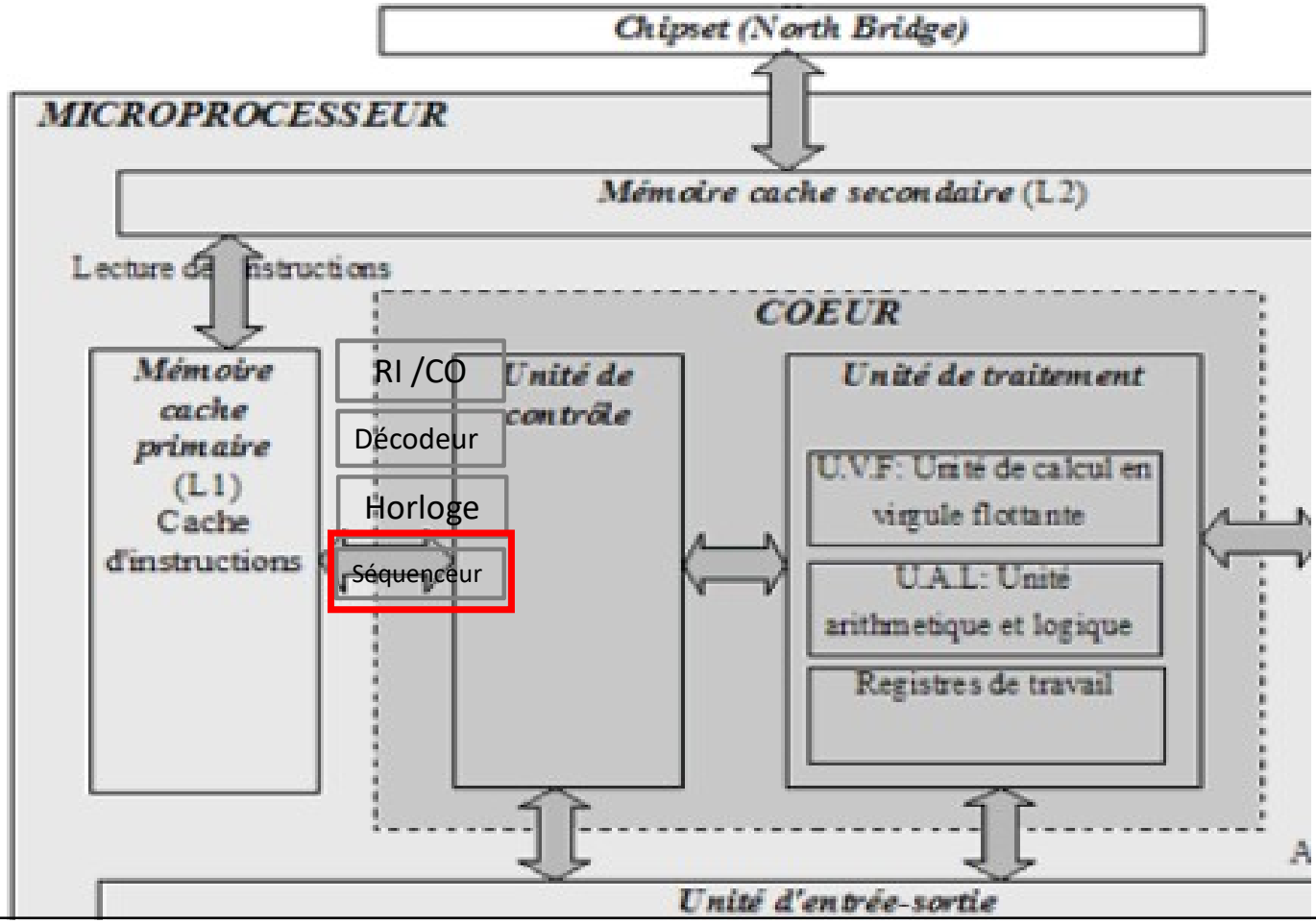


Horloge

Définition

Définit le cycle de base : cycle machine, Utilisée pour synchroniser chaque étape des cycles de recherche et d'exécution,

Le temps d'exécution d'une instruction n'est généralement pas égal au cycle machine, car il faut plusieurs cycles machine pour pouvoir séquentiellement récupérer l'instruction, la décoder et récupérer les données qu'elle doit traiter.
Exemple : sur les microcontrôleurs PIC, il faut 4 cycles d'horloge (machine) pour exécuter une instruction.


Chipset (North Bridge)
MICROPROCESSEUR
Mémoire cache secondaire (L2)
Lecture des instructions
COEUR
Mémoire cache primaire (L1) Cache d'instructions
RI /CO
Décodeur
Horloge
Séquenceur
Unité de contrôle
Unité de traitement
U.V.F: Unité de calcul en virgule flottante
U.A.L: Unité arithmetique et logique
Registres de travail
Unité d'entrée-sortie

Séquenceur

Définition

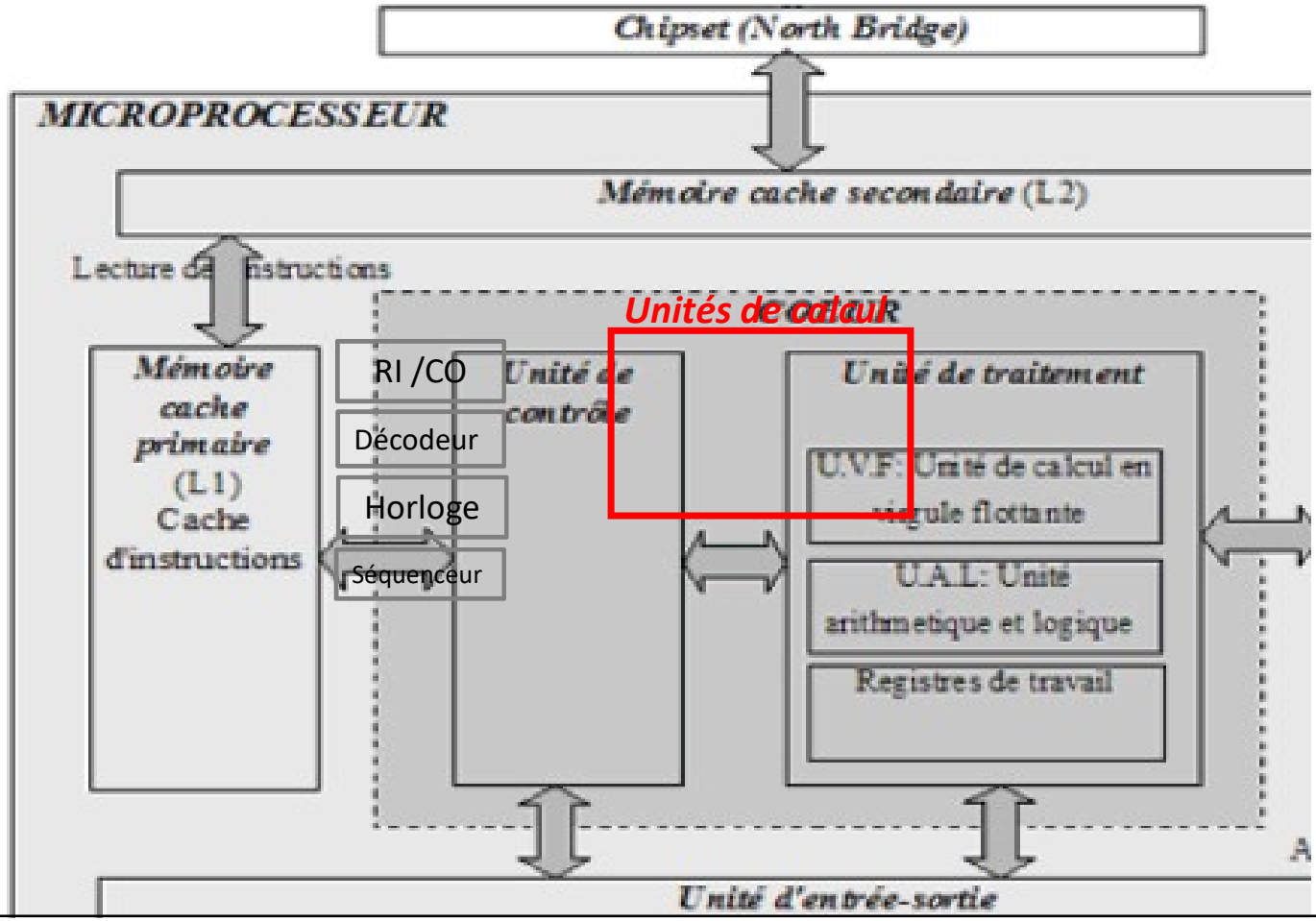
Automate générant les signaux de commande contrôlant les différentes unités,

Il existe 2 façons de réaliser cette automate :

- Séquenceur câblé,
- Séquenceur micro-programmé (firmware),

Avantages :

- Câblé : un peu plus rapide,
- Micro-programmé : plus souple et plus simple à réaliser,


Chipset (North Bridge)
MICROPROCESSEUR
Mémoire cache secondaire (L2)
Lecture des instructions
Unités de calcul
Mémoire cache primaire (L1) Cache d'instructions
RI /CO
Décodeur
Horloge
Séquenceur
Unité de contrôle
Unité de traitement
U.V.F: Unité de calcul en virgule flottante
U.A.L: Unité arithmetique et logique
Registres de travail
Unité d'entrée-sortie

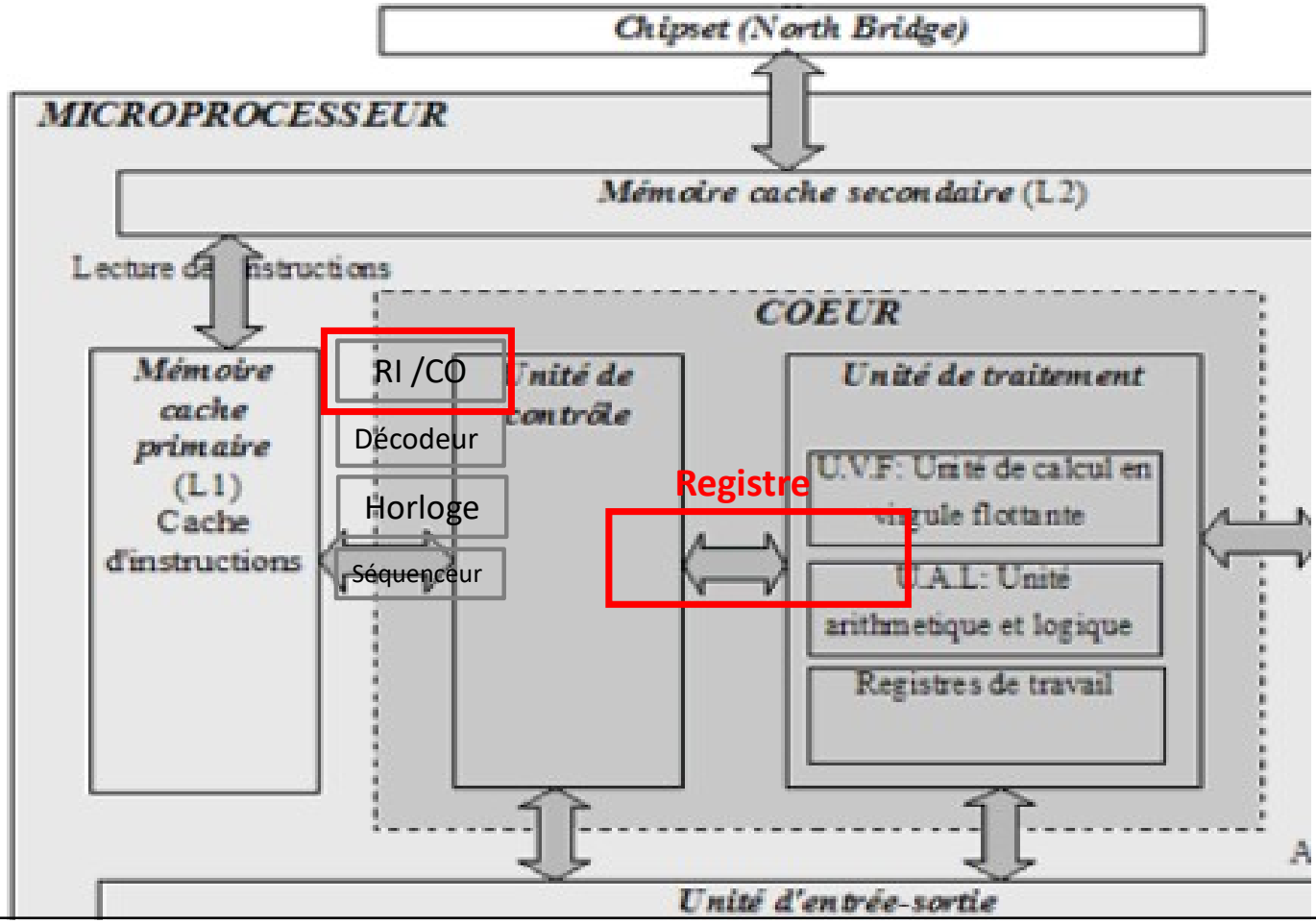
Unités de calcul

Définition

Un processeur est avant tout un organe de calcul integrant des unités de calcul, pouvant être de plusieurs types :

- Arithmétique et logique (ALU)
- Flottant (FPU)
- Autres

Un processeur peut intégrer une ou plusieurs unités de type identique ou différent. Par exemple, l'Athlon 64 d'AMD embarque 3 ALU et 3 FPU (calculs en parallèle le possible)


Chipset (North Bridge)
MICROPROCESSEUR
Mémoire cache secondaire (L2)
Lecture des instructions
COEUR
Mémoire cache primaire (L1) Cache d'instructions
RI /CO
Décodeur
Horloge
Séquenceur
Unité de contrôle
Unité de traitement
Registre
U.V.F: Unité de calcul en virgule flottante
U.A.L: Unité arithmetique et logique
Registres de travail
Unité d'entrée-sortie

Registre

Définition

Mot mémoire interne au processeur,

Plusieurs types :

- ***Registres de fonctionnement*** :
  - Compteur Ordinal (CO), Registre Instruction (RI)...
  - Accumulateur (interne à l'ALU)

- ***Registres généraux*** :Servent à stocker des valeurs souvent utilisées ou des résultats intermédiaires sans passer par la mémoire,

- ***Registres de pile (SP : Stack Pointer)***

Registre

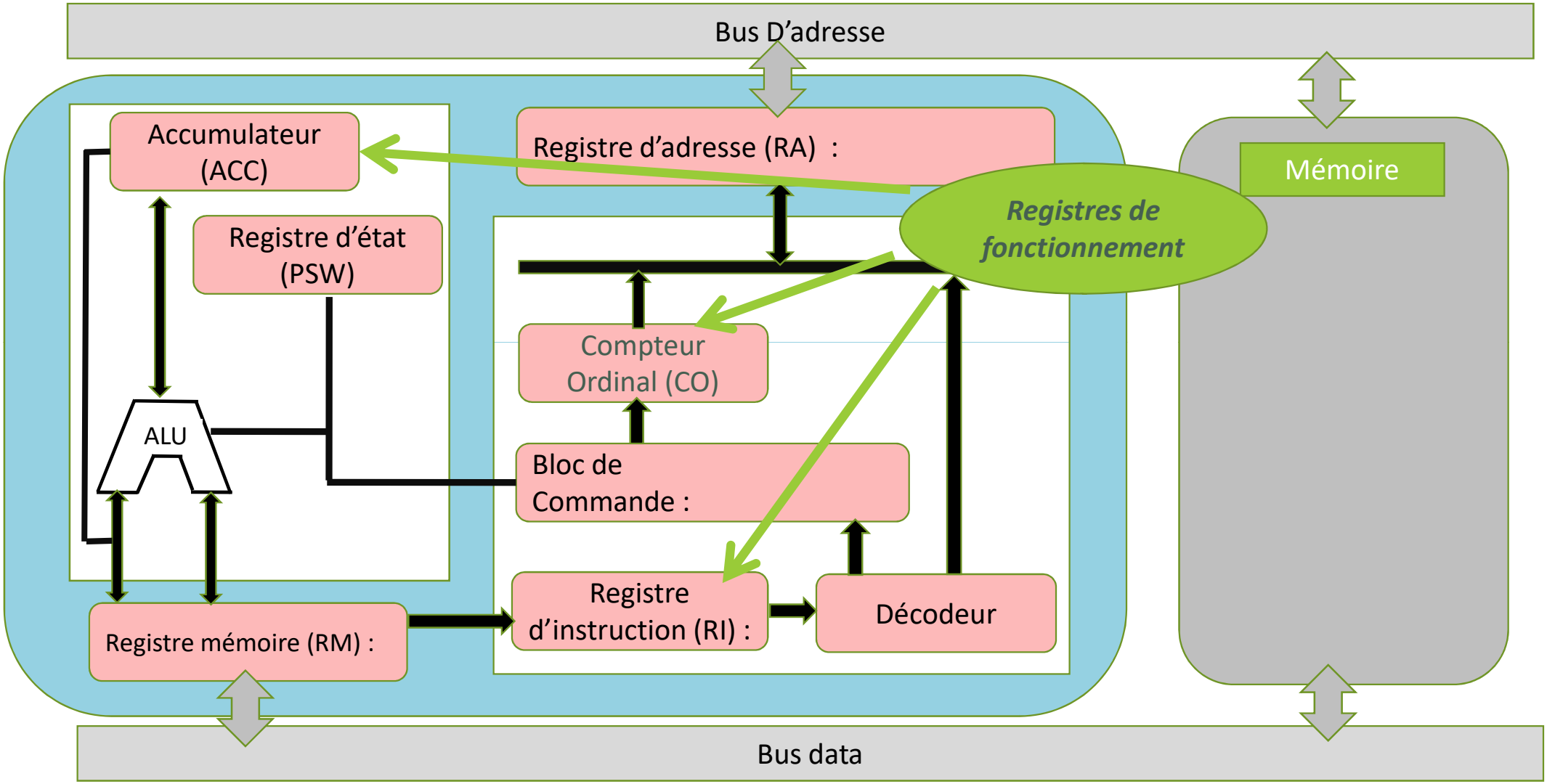
- ***Registres d'état*** (***PSW : Program Status Word)*** :Ensemble de bits représentant chacun un état particulier (drapeau ou flag),
  - C : dépassement de capacité après un calcul de l'ALU
  - Z : résultat de l'opération est égal à 0

- ***Registre d'Adresse (RA) :*** registre contenant l'adresse du mot à accéder en mémoire

- ***Registre Mémoire (RM) :*** registre contenant le mot lu ou à écrire en mémoire

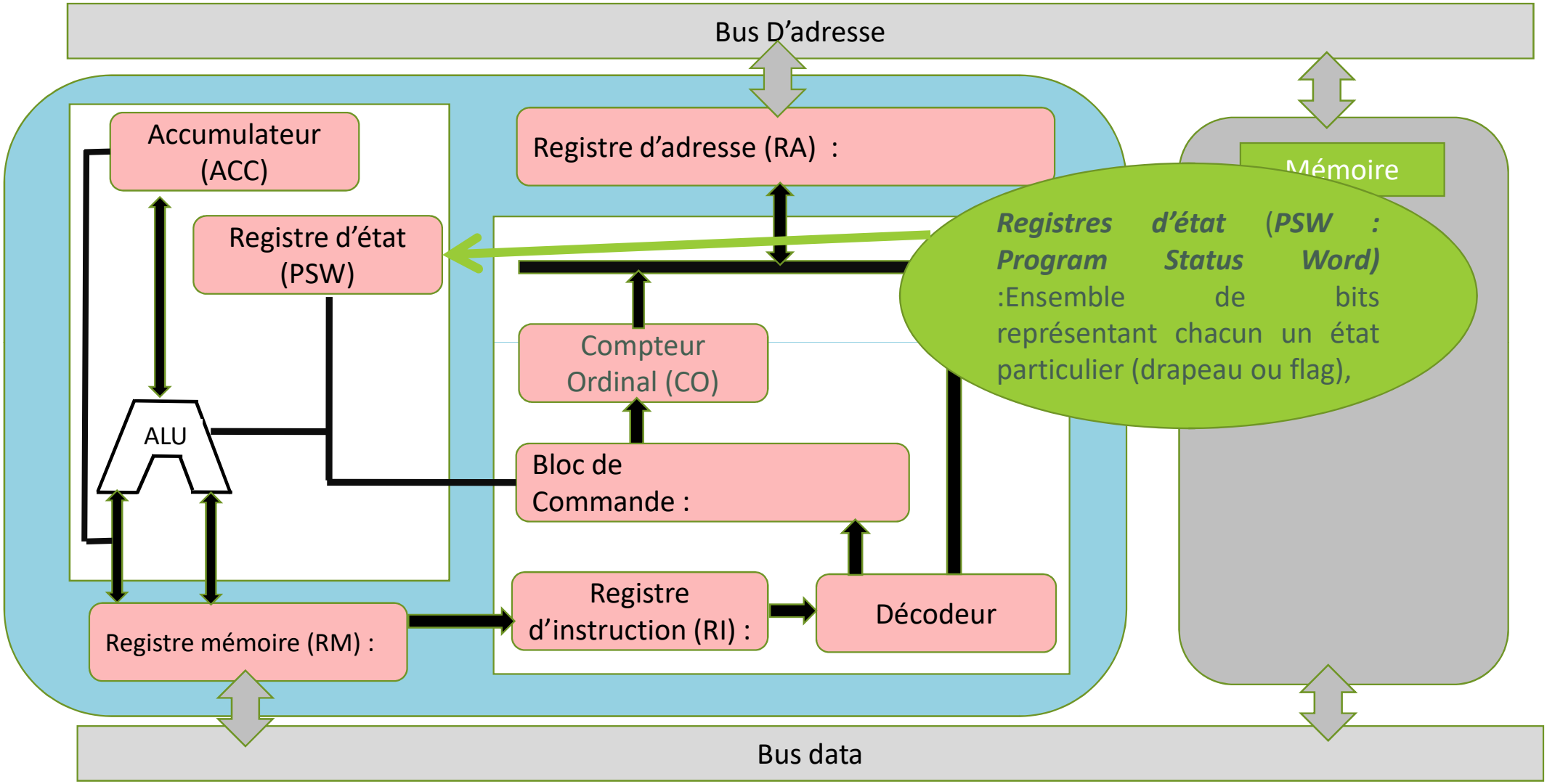
Registre

Mr. MEGNAFI Hicham (ESSA-Tlemcen)



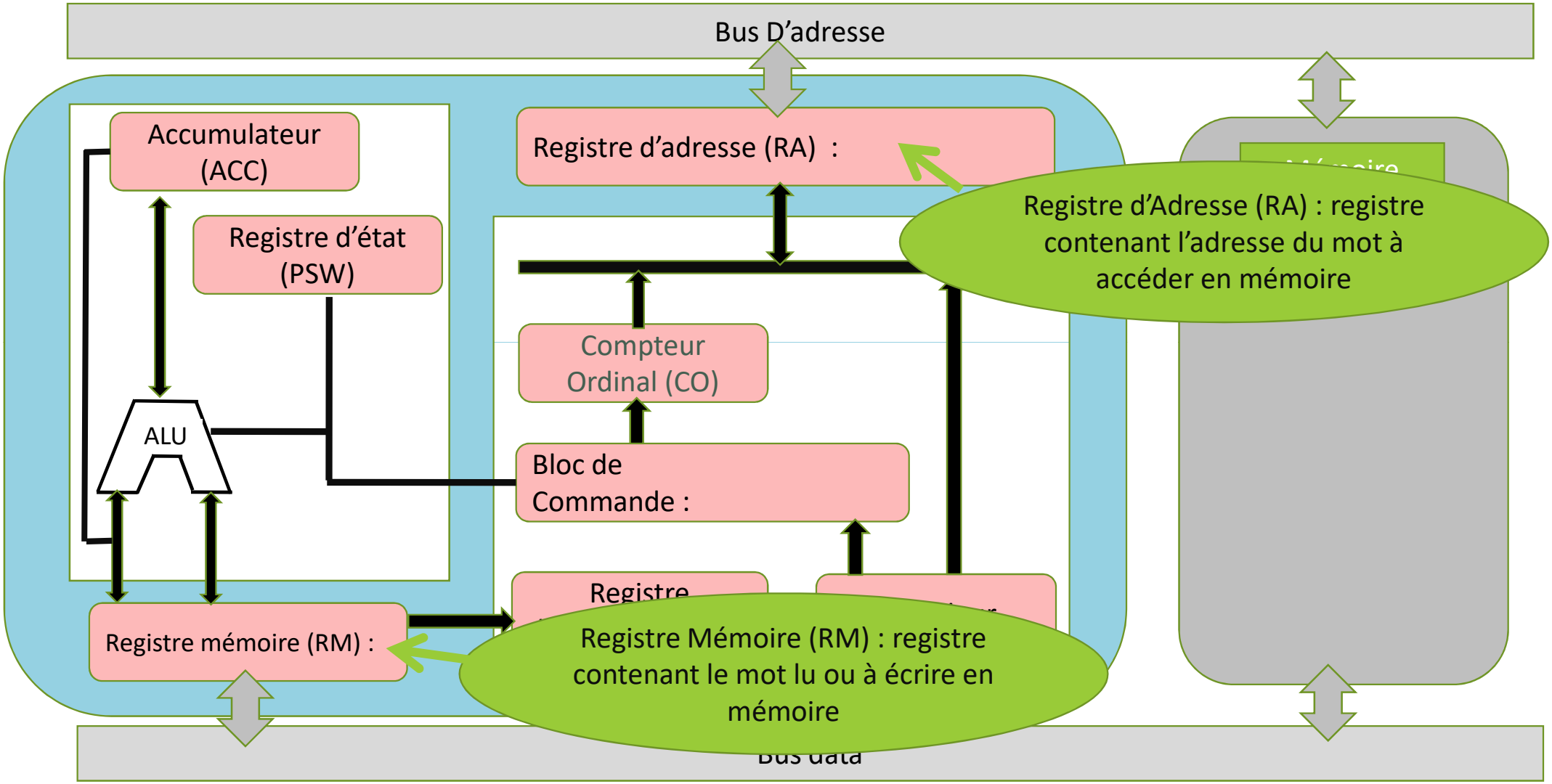
Registre

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

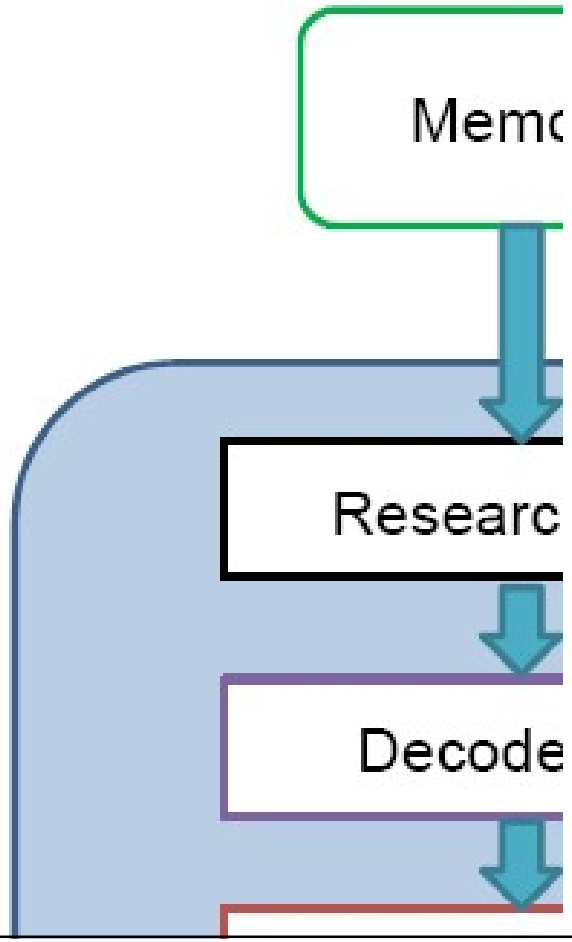


Registre

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

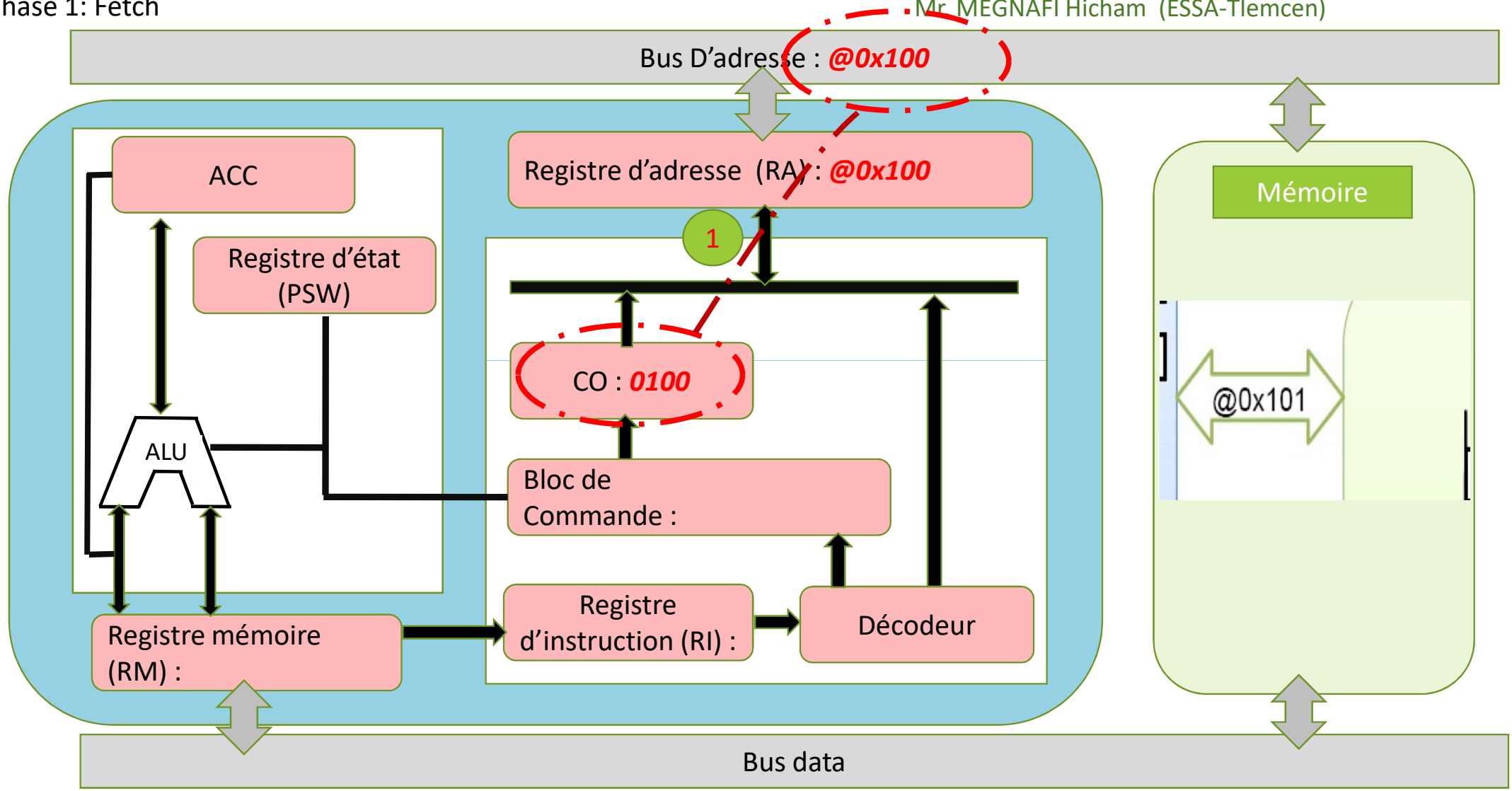


How a µP work?

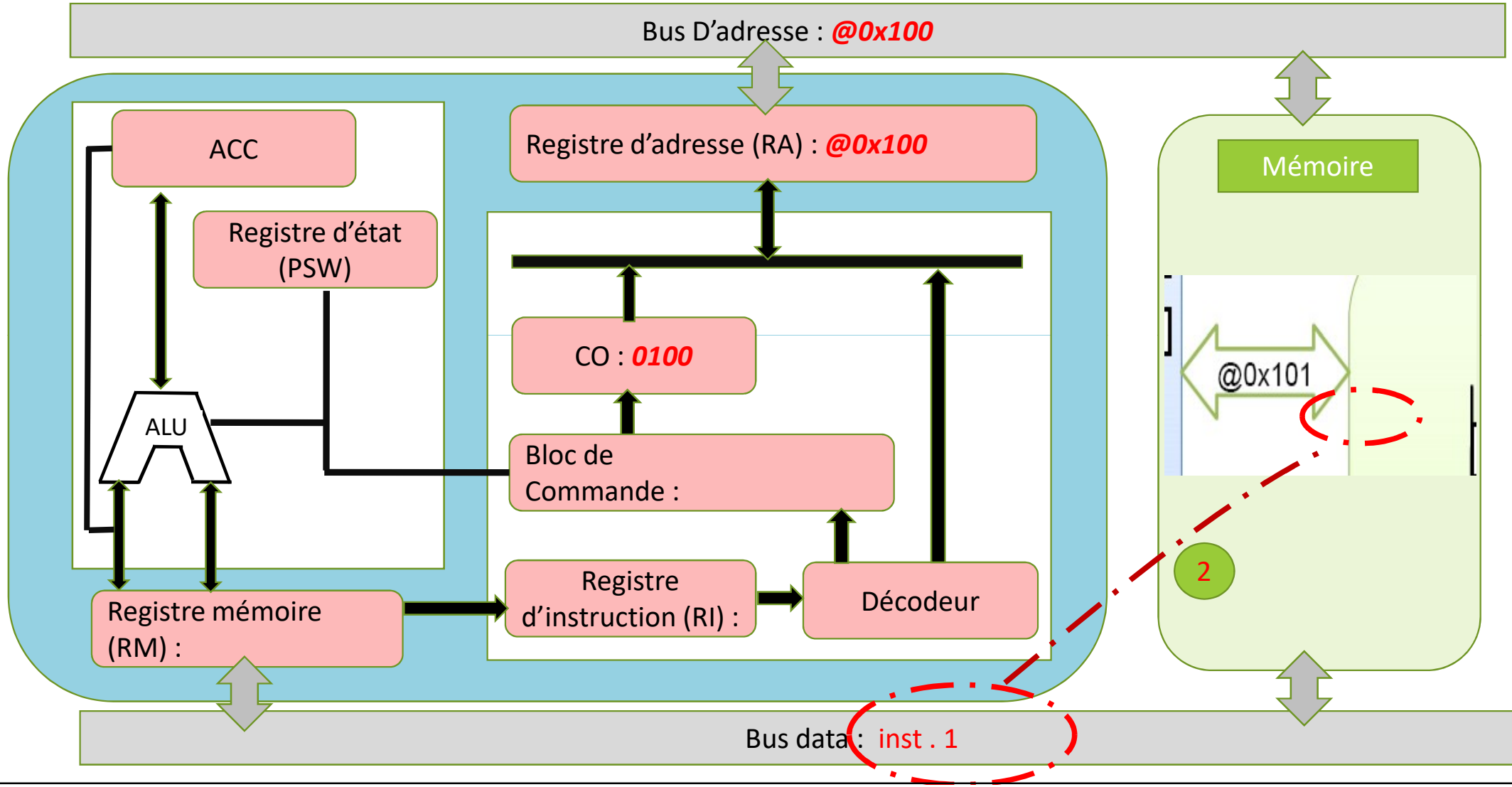


Phase 1: Fetch

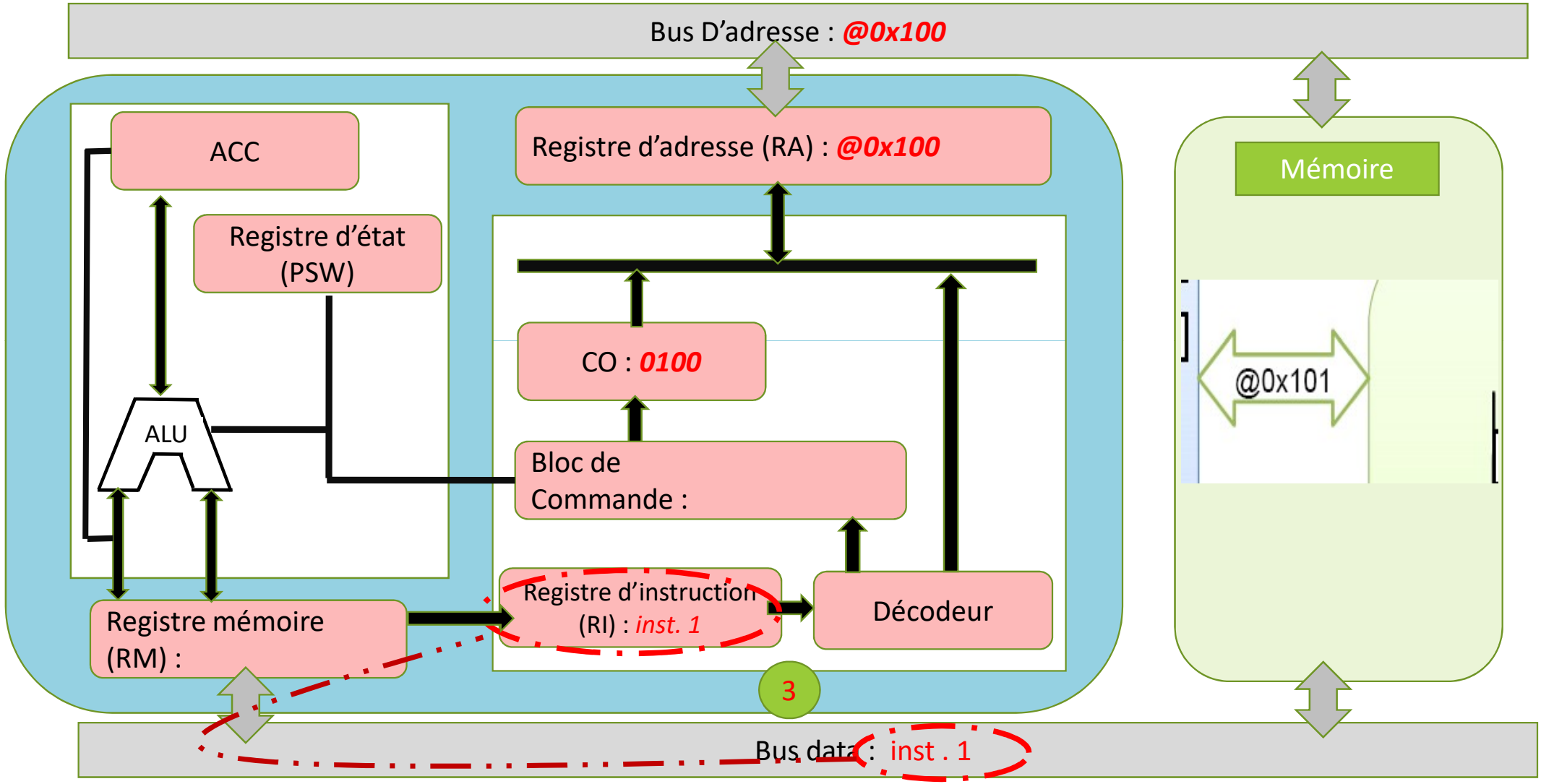
Mr. MEGNAFI Hicham (ESSA-Tlemcen)



Phase 1: Fetch

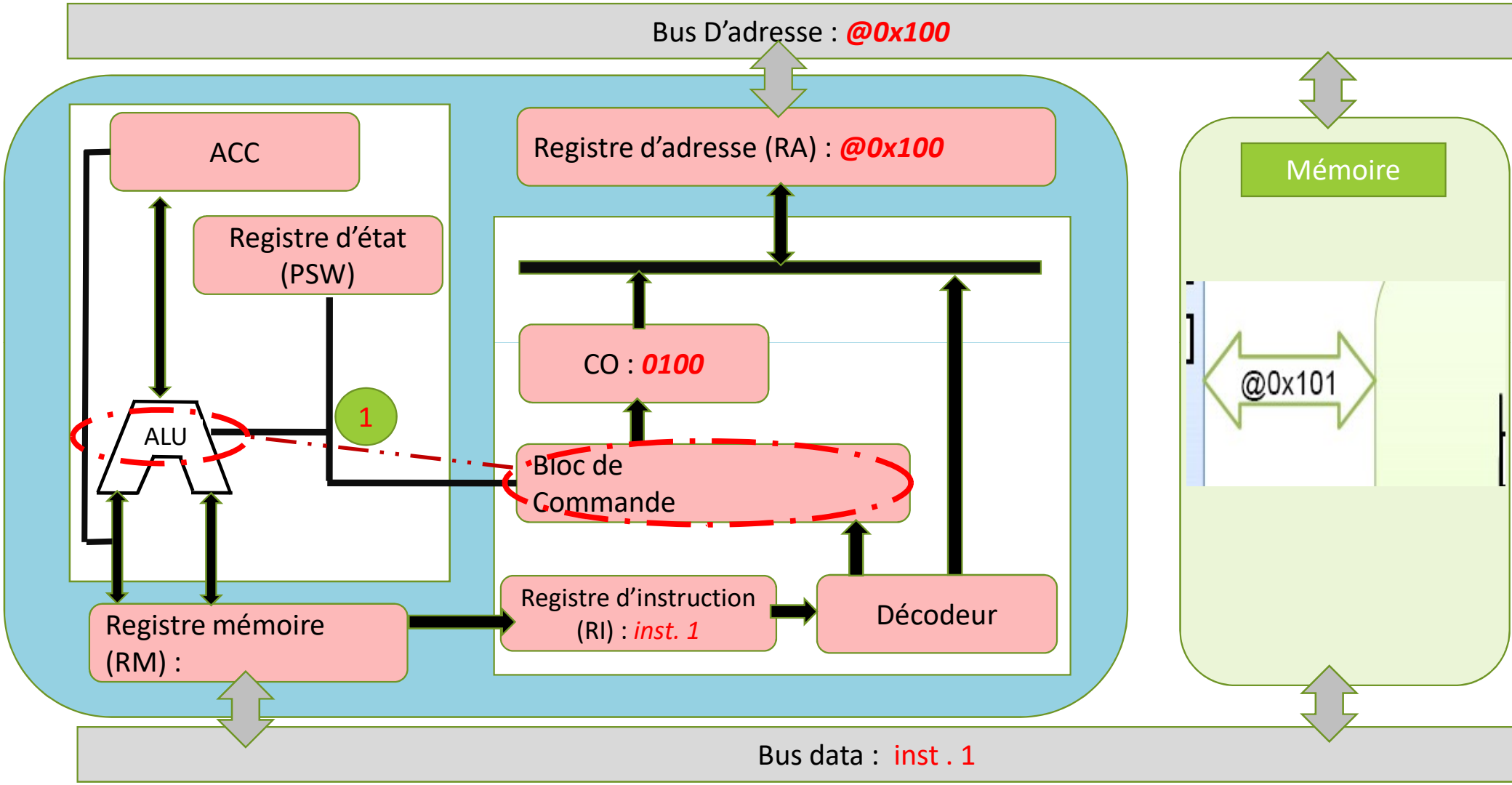


Phase 1: Fetch



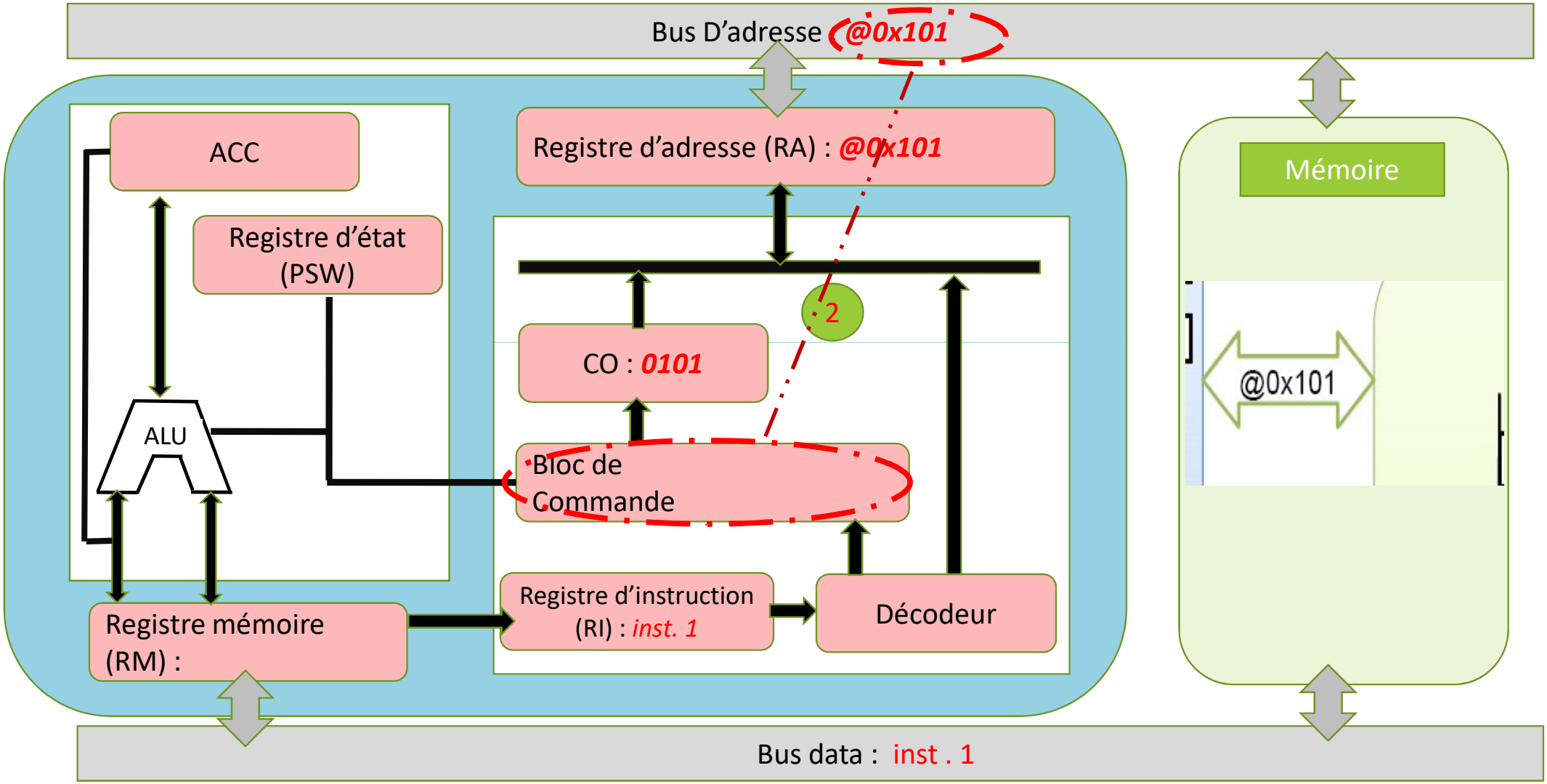
Phase 2: Decode and operand search

Mr. MEGNAFI Hicham (ESSA-Tlemcen)



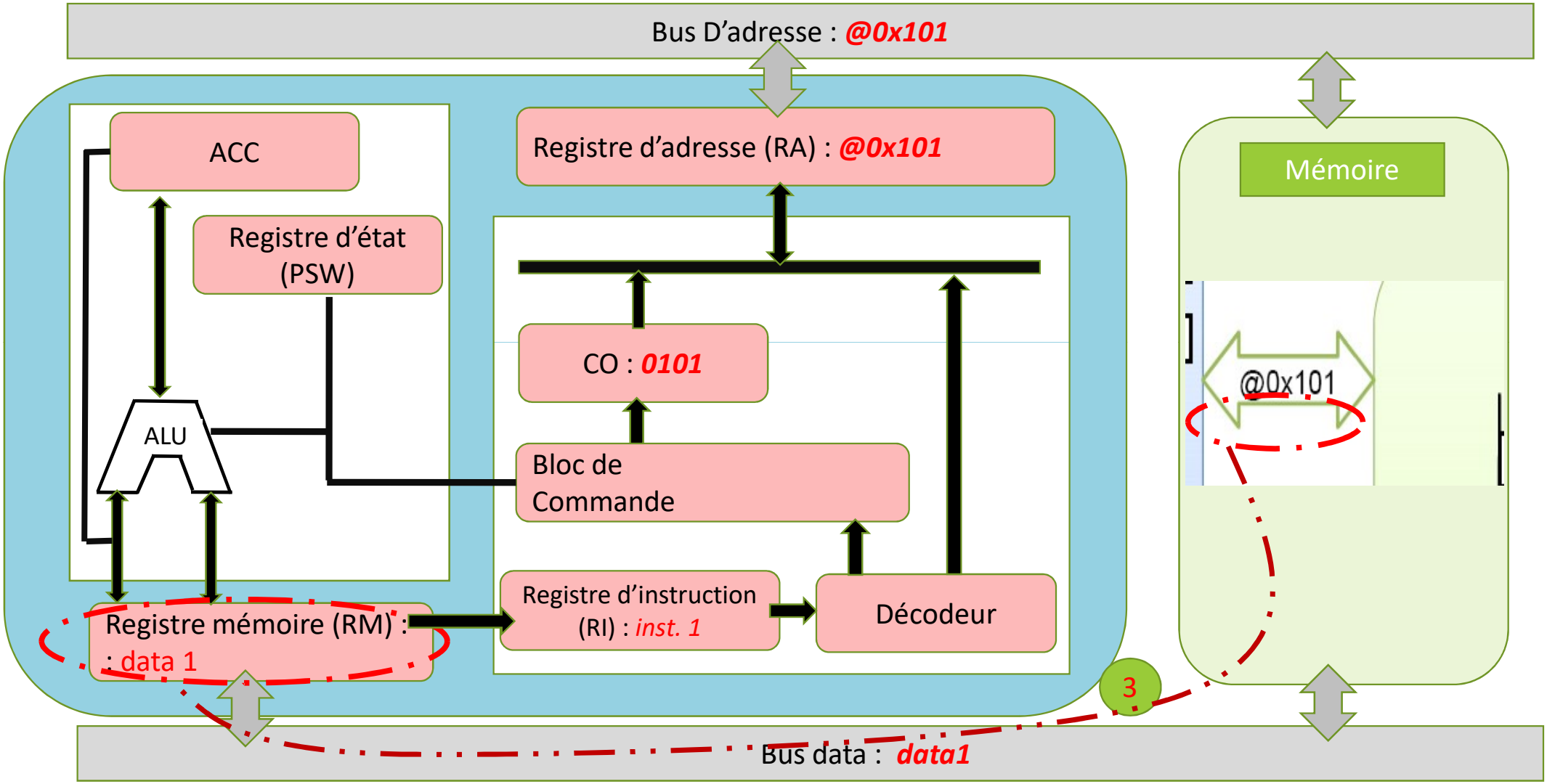
Phase 2: Decode and operand search

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

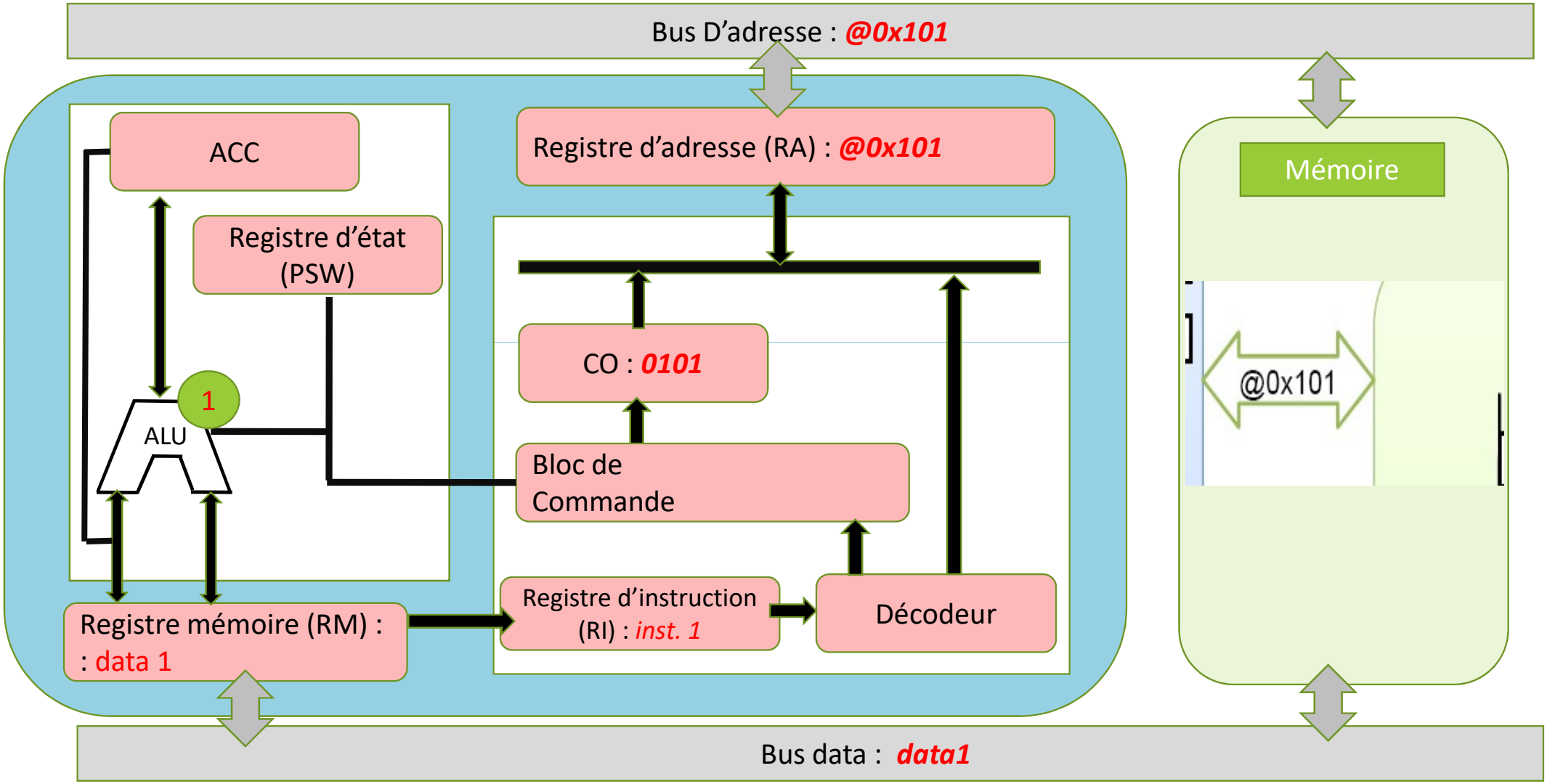


Phase 2: Decode and operand search

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

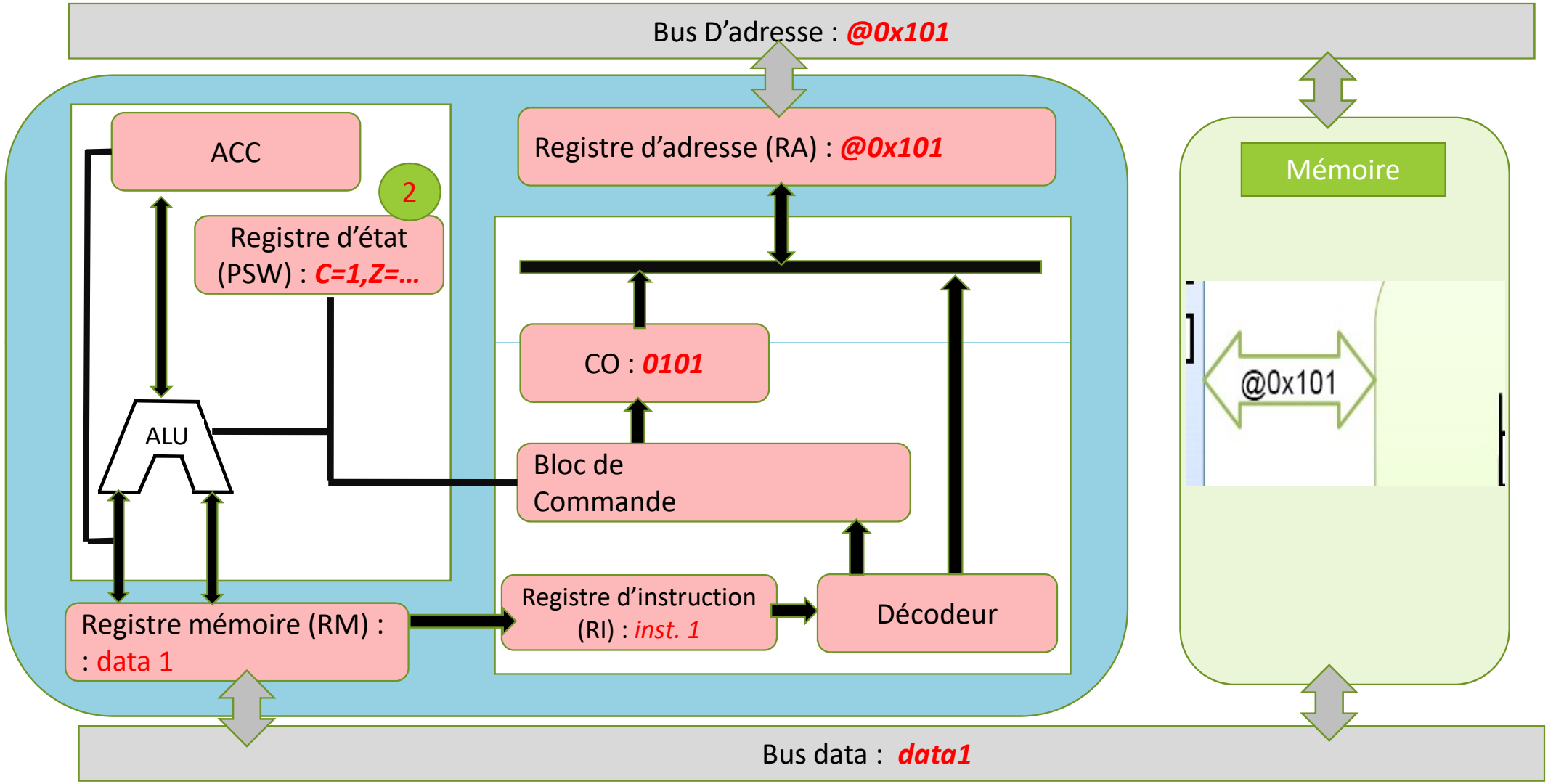


Phase 3: Execution



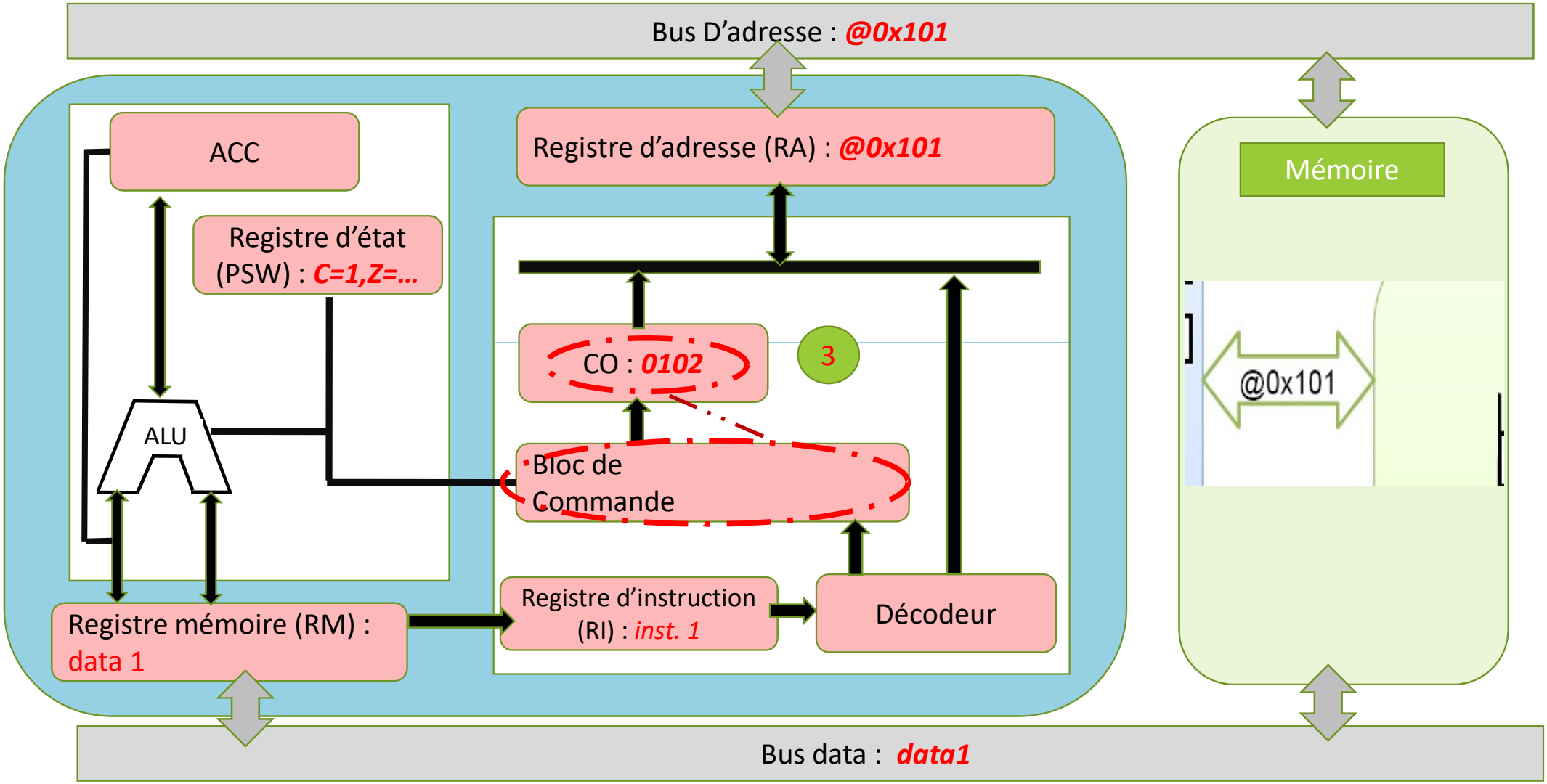
Phase 3: Execution

Mr. MEGNAFI Hicham (ESSA-Tlemcen)



Phase 3: Execution

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

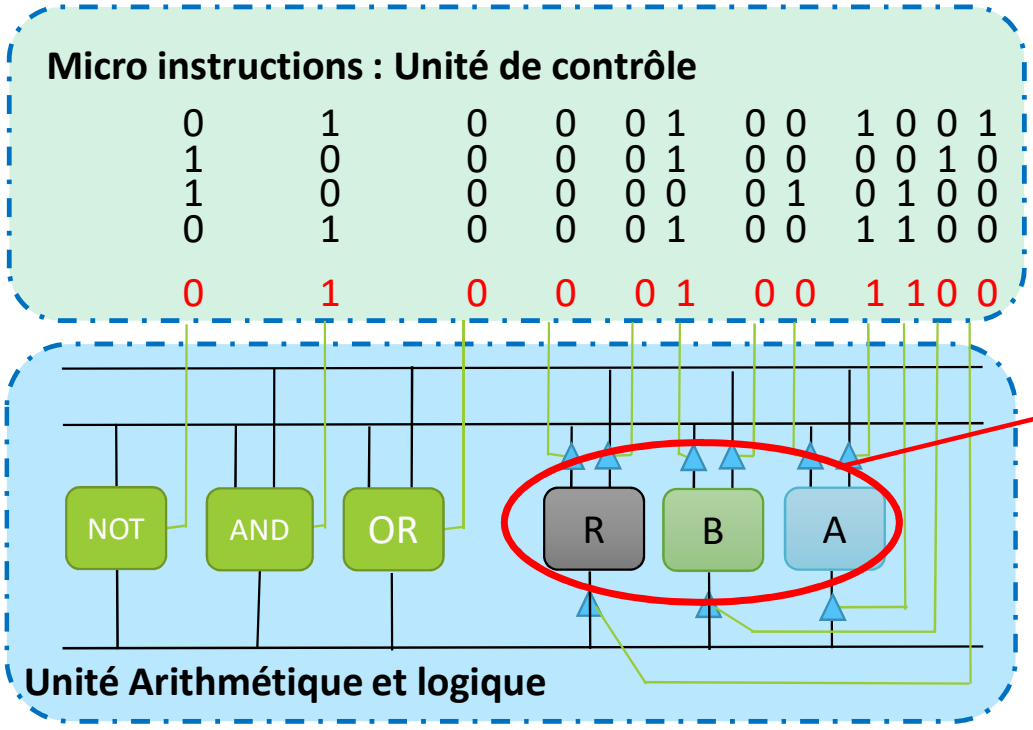


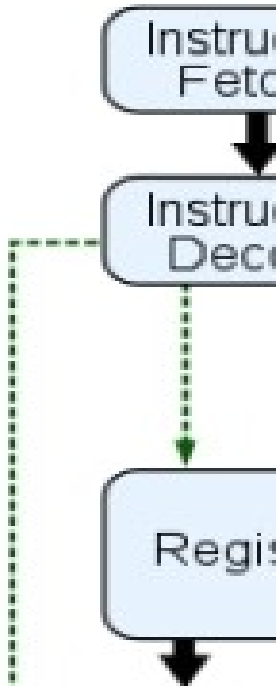
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

jeux d'instructions (1/5)

Définition : C'est l'opération élémentaire qu'un programme demande à un processeur d'effectuer.,

called processor) is a machine





jeux d'instructions (2/5)

- ***Les instructions sont codées en binaire.***
- ***Une instruction est composée de deux éléments :***
  - Le code opération : C'est un code binaire qui correspond à l'action à effectuer par le processeur
  - Le champ opérande : Donnée ou bien adresse de la donnée.
- ***La taille d'une instruction peut varier, elle est généralement de quelques octets (1 à 8), elle dépend également de l'architecture du processeur.***

jeux d'instructions : Exemple d'instruction (3/5)

Instruction Addition
Accumulateur = Accumulateur + Opérande
Correspond à l'instruction ADD A,#2

Instruction (16 bits)	
Code opératoire (5 bits)	Champ opérande
ADD A	#2

Figure : Exemple d'instruction d'addition

Cette instruction est comprise par le processeur par le mot binaire :
11001 000 0000 0010 (code machine)

jeux d'instructions : Exemple d'instruction (4/5)

Remarque : Code opératoire : Puisque sa valeur numérique n'a pas de sens pour les humains, le programmeur utilise une abréviation désignant le code opération fourni par le langage assembleur pour ce processeur appelée *mnémonique*.

jeux d'instructions (5/5)

Types d'instructions :

- **Traitement**
  - opérations arithmétiques et logiques
- **Transferts des données avec la mémoire**
  - load, store
- **Contrôle**
  - Branchements
  - Branchements aux sous programmes(Sauvegarde automatique de l'adresse de retour en « PILE »)
- **Système**
  - Interruptions logicielles : Appel de fonctions de l '«operating system »
- **Coprocesseur**
  - Instructions spécifiques à un opérateur extérieur « coprocesseur »

Types de jeux d'instructions (1/8)

Deux grands types de jeux d'instructions

- **CISC : Complex Instruction Set Computing**
  - Exemples : familles x86 (Intel / AMD)
  - Fonctionne en modèle mémoire-mémoire généralement
- **RISC : Reduced Instruction Set Computing**
  - Exemples : Sun Sparc, MicroChip PIC
  - Fonctionne en modèle mémoire-registre généralement

Types de jeux d'instructions : CISC (2/8)

Complex Instruction Set Computing (CISC) Le passé : Processeur CISC

- **L'accès à la mémoire extérieure est relativement lent,**
- **L'exécution d'une instruction nécessite plusieurs étapes**

Figure : Etapes d'exécution d'une instruction

Dans les années 1970, après la naissance des premiers microprocesseurs, les accès mémoires étaient relativement lents et le processeur disposait de plusieurs cycles d'horloge (au moins 8 pour le 68000) pour aller chercher, décoder et exécuter l'instruction.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Types de jeux d'instructions : **CISC (3/8)**

Complex Instruction Set Computing (CISC)

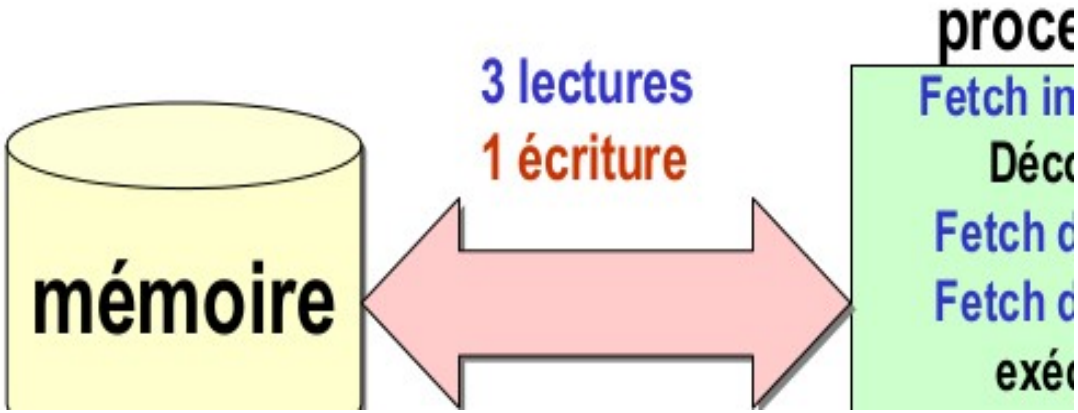


Figure : Modèle mémoire-mémoire

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Types de jeux d'instructions : RISC (4/8)

Processeur RISC : Reduced Instruction Set Computing

- **Utiliser des registres, de la mémoire locale rapide ou « cache » pour éviter d'attendre,**
- **Paralléliser les instructions : techniques pipeline,**
- **Instructions simples au format fixe,**
- **2 instructions pour l'accès à la mémoire : LOAD STORE,**

Dans les années 80, cette approche devient possible grâce aux progrès technologiques Le gain en performance doit tendre vers « une instruction en 1 cycle d'horloge ».

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Types de jeux d'instructions : RISC (5/8)

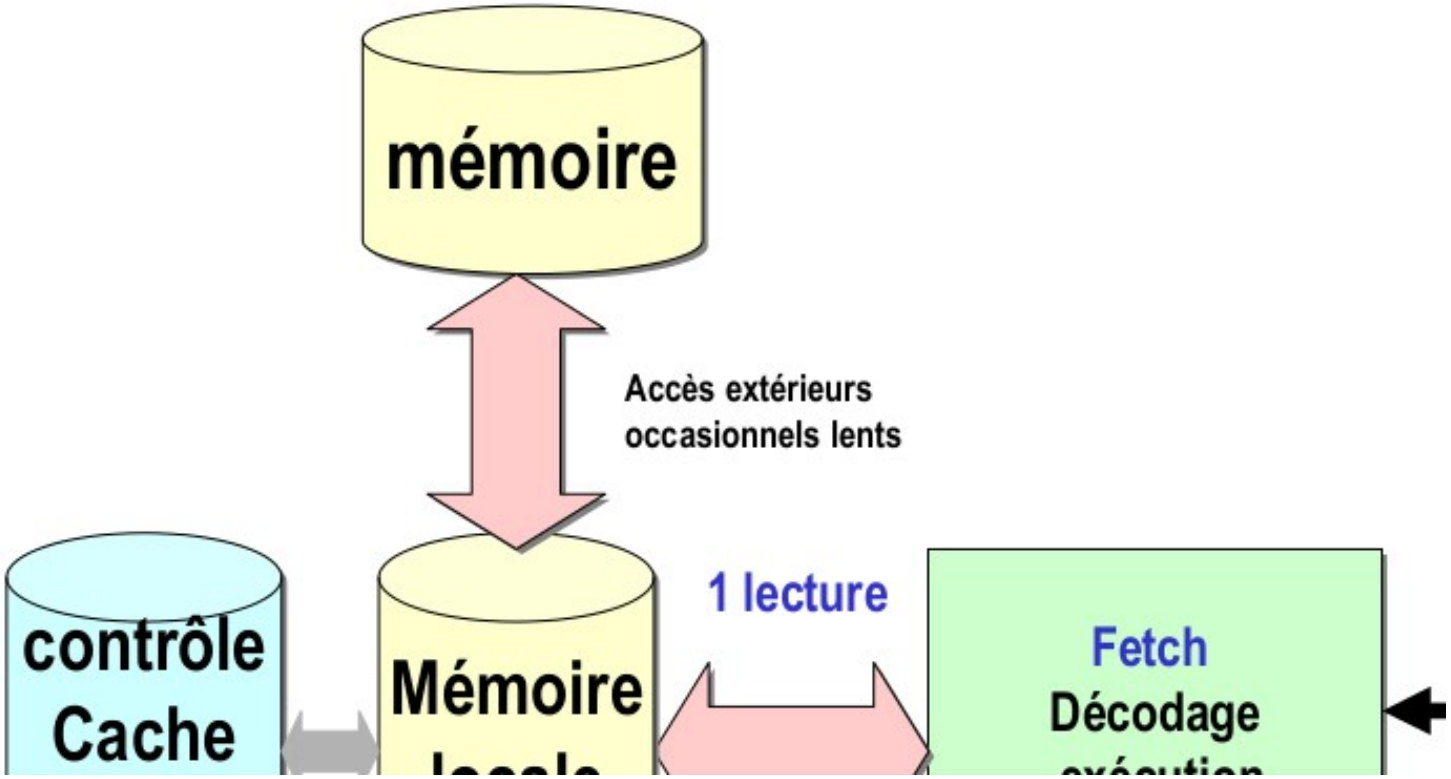


Figure : Modèle mémoire-registre

Types de jeux d'instructions : RISC (6/8)

Dans la fin des années 1980, il est devenu plus facile d'intégrer de la mémoire rapide sous forme de bancs de registre, mémoire locale ou mémoire cache au sein même du processeur. Le micro-séquencement des instructions ne se justifiait plus mais les instructions devait être assez simples pour être exécutées le plus vite possible, c'est à dire en 1 cycle. Le concept "Reduced Instruction Set Computer" était né.

D'autre part les calculs effectués sur des registres permettent de simplifier l'architecture et de faciliter le respect des performances. Pour charger ces registres ou sauver leur valeurs en mémoire, il suffit d'avoir respectivement 2 instructions simples LOAD et STORE.

Il n'en demeure pas moins qu'il faut toujours les mêmes étapes pour exécuter une instruction RISC : - Aller la chercher en mémoire,

- La décoder,
- L'exécuter

Une façon simple de "paralléliser" ces étapes est de les mettre en pipeline comme il va être vu par la suite.

Types de jeux d'instructions (7/8)

Exemple d'instruction CISC et RISC Faire la somme du registre W et de la valeur à l'adresse 20 en mémoire et placer le résultat à l'adresse 30 :

- En CISC
  - Une seule opération nécessaire : ADD W, @20, @30
- En RISC (exemple PIC 16F84)
  - MOVLW 0x54
  - ADDWF 0x20, 0
  - MOVWF 0x30
  - 3 instructions nécessaires

Types de jeux d'instructions (8/8)

Comparaison entre RISC et CISC

RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
- Instruction simple: 1 instruction = 1 cycle - format d'instruction fixe - compilateur complexe - Structure de pipeline	- Instructions complexes consommant plus d'un cycle - Format d'instruction variable - Compilateur simple - Beaucoup de mode d'adressage

Utilisation de pipeline (1/6)

Un pipeline (ou chaîne de traitement), est l'élément d'un processeur dans lequel l'exécution des instructions est découpée en plusieurs étapes. Avec un pipeline, le processeur peut commencer à exécuter une nouvelle instruction sans attendre que la précédente soit terminée. Chacune des étapes du pipeline est implémentée par un circuit integré indépendant, appelé étage. Le nombre d'étages d'un pipeline est appelé sa profondeur.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Utilisation de pipeline (2/6)

Les principes :

- Une instruction est divisée en k étapes indépendantes
- Les étapes de plusieurs instructions s'exécutent en parallèle
- Si une étape dure t unités de temps alors pour n instructions :
  - Sans pipeline : t * k * n unités de temps
  - Avec pipeline : t * k + t * (n-1) unités de temps

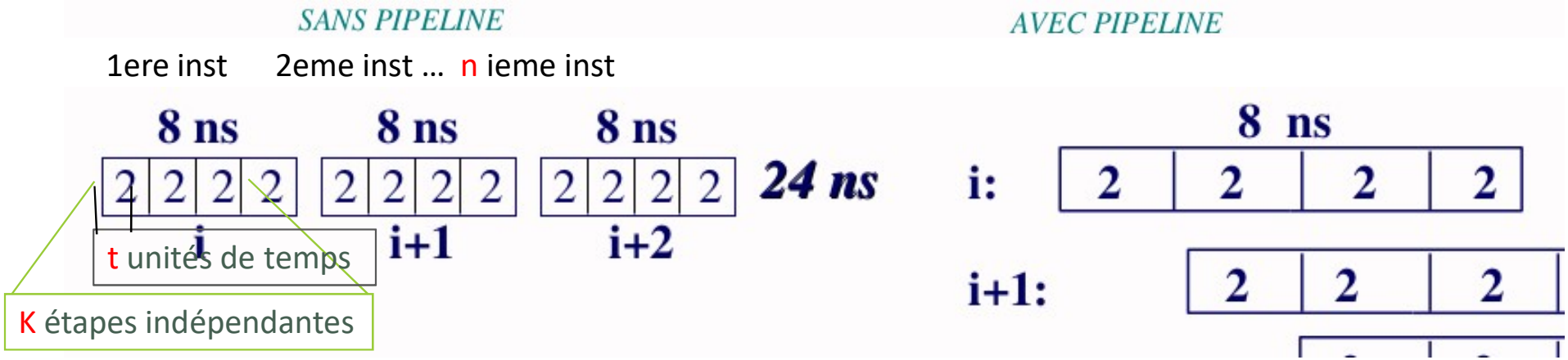


Figure : Temps d'exécution d'une instruction sans et avec pipeline

Utilisation de pipeline (3/6)

Contraintes pour que le pipeline soit efficace :

- Chaque étape a la même durée : pas d'embouteillage !
- Chaque instruction passe par les mêmes étapes : contrôle simplifié, pas de trou dans le pipeline !

Utilisation de pipeline (4/6)

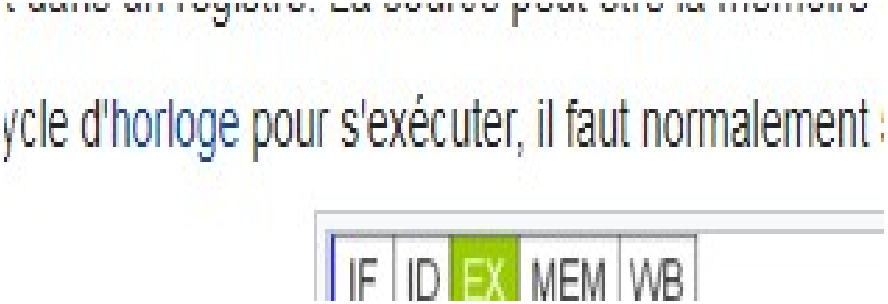
Étages de base (Pipeline RISC classique):

Avec ce pipeline, 5 étapes sont nécessaires pour traiter une instruction :

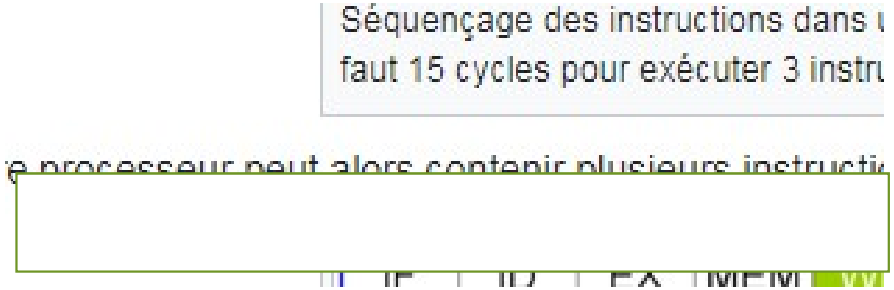
1. **IF (Instruction Fetch)** charge l'instruction à exécuter dans le pipeline.
2. **ID(Instruction Decode)** décode l'instruction et adresse les registres.
3. **EX (Execute)** exécute l'instruction (par la ou les unités arithmétiques et logiques).
4. **MEM (Memory),** dénote un transfert depuis un registre vers la mémoire dans le cas d'une instruction du type STORE (accès en écriture) et de la mémoire vers un registre dans le cas d'un LOAD (accès en lecture).
5. **WB (Write Back)** stocke le résultat dans un registre. La source peut être la mémoire ou bien un registre.

Utilisation de pipeline (5/6)

En supposant que chaque étape met 1 cycle d'horloge pour s'exécuter.



Séquençage des instructions dans un processeur sans pipeline. Il faut 15 cycles pour exécuter 3 instructions.



Séquençage des instructions dans un processeur doté d'un pipeline à 5 étages. Il faut 7 cycles pour exécuter 3 instructions. À t = 5, tous les étages du pipeline sont sollicités

Utilisation de pipeline (6/6)

Exemples de pipeline :

- Intel Pentium 4 : 20
- Intel Pentium II: 14
- AMD Athlon : 12
- Motorola PowerPC G4: 7

TD

Question 1

Qu'est-ce que le pointeur d'instruction ?

- ❑Le registre contenant l'instruction en cours.
- ❑Le registre contenant l'adresse de l'instruction en cours.
- ❑Le registre contenant l'adresse de l'instruction suivante.

Question 2

Un microprocesseur dit RISC a pour principales caractéristiques ?

- ❑Un set d'instructions étendu et un nombre de cycles par instruction élevé
- ❑Des set d'instructions et nombre de cycles par instruction réduits
- ❑Un set d'instructions réduit et un nombre de cycles par instruction élevé

Question 3

Un Quels sont les éléments constituant un microprocesseur ?

Question 3

Un Quels sont les éléments constituant un microprocesseur ?

- une unité de contrôle (UC).
- une unité arithmétique et logique (UAL).
- Les registres.
- mémoire cache.

Question 4

L'élément du processeur spécialisé pour les calculs est ?

- ❑ LLC.
- ❑ ALU.
- ❑ UCC.

Question 5

Dans un ordinateur, les données sont présentées par un signal électrique dela forme

- ❑Analogique.
- ❑Numérique.
- ❑Analogique et numérique.

Question 7

Quel registre permet de stocker les données en cours de traitement par l'UAL ?

- ❑ Instruction
- ❑ Etat
- ❑ Accumulateur

Question 8

Quel est la fonction du pipeline ?

- ❑Il permet la transition de données à virgule.
- ❑Il permet la récuperation de données.
- ❑Il permet d'assembler temporellement les traitements a effectuer.

Plan de Cours

1. Projets réalisés
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. ***Type des mémoires***
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

5.1 Mémoire
5.2 Mémoire Chronogramme de lecture/écriture
5.3 Caractéristique de la mémoire
5.4 Classification des mémoires
a) Mémoires Morts
- ***ROM,EPROM,EEPROM,FLASH***

b) Mémoires Vives
- ***Static ,Dynamic RAM***

5.5 Pyramide de la mémoire
5.6 mémoire séquentielle, LIFO, FIFO

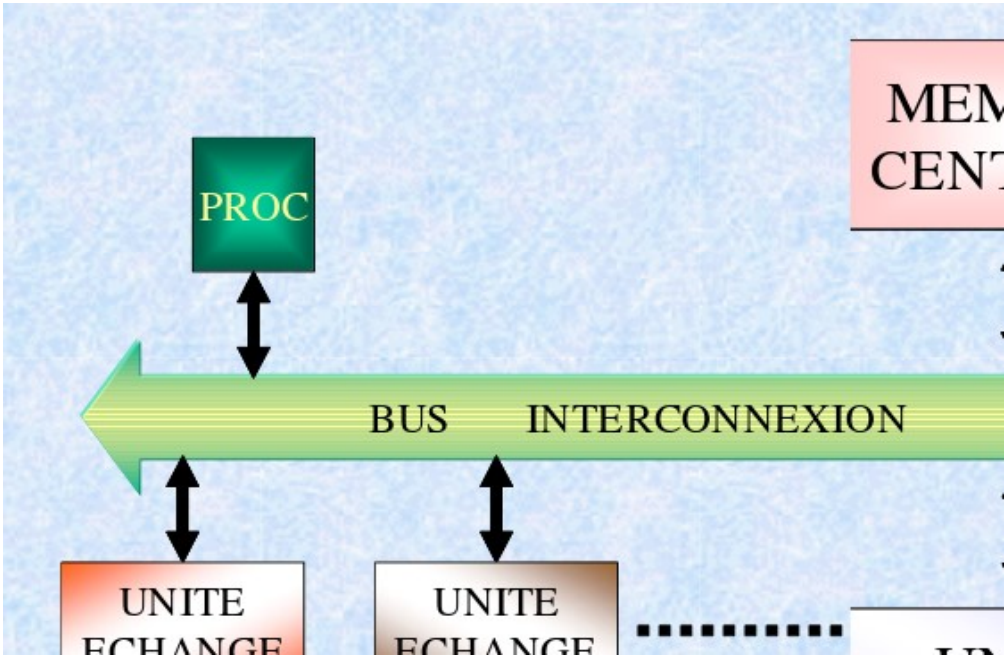
Mémoire

Définition:

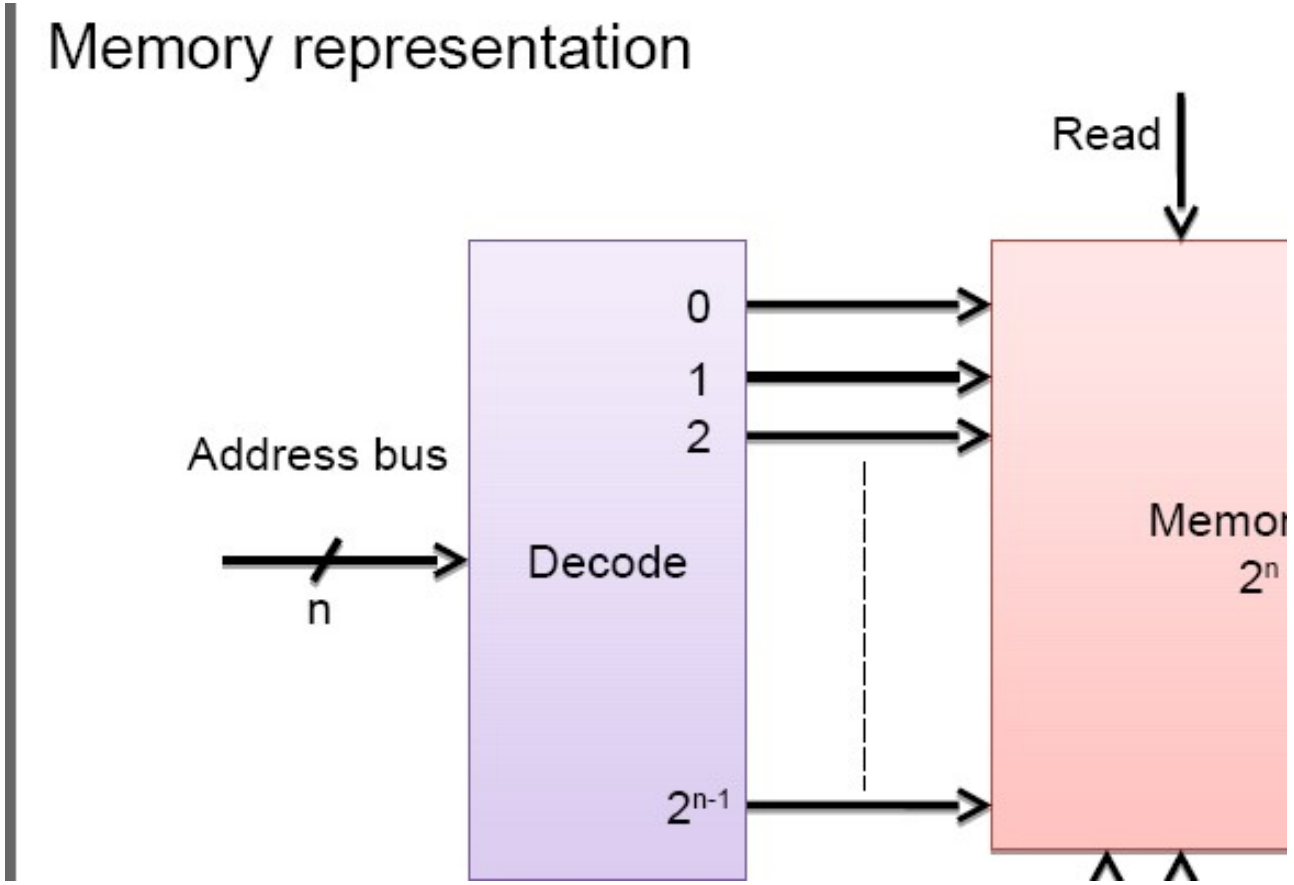
- Espace destiné a recevoir, conserver et restituer des informations à traiter,
- Tout composant électronique capable de stocker temporairement des données

Mémoire

- Assemblage de Transistors et condensateurs
- Circuit décodeur d'adresse
- Exploitées pour construire les :
  - Registres du processeur,
  - la mémoire centrale,
  - les ports d'Entrées / Sorties (Unité d'Echange)



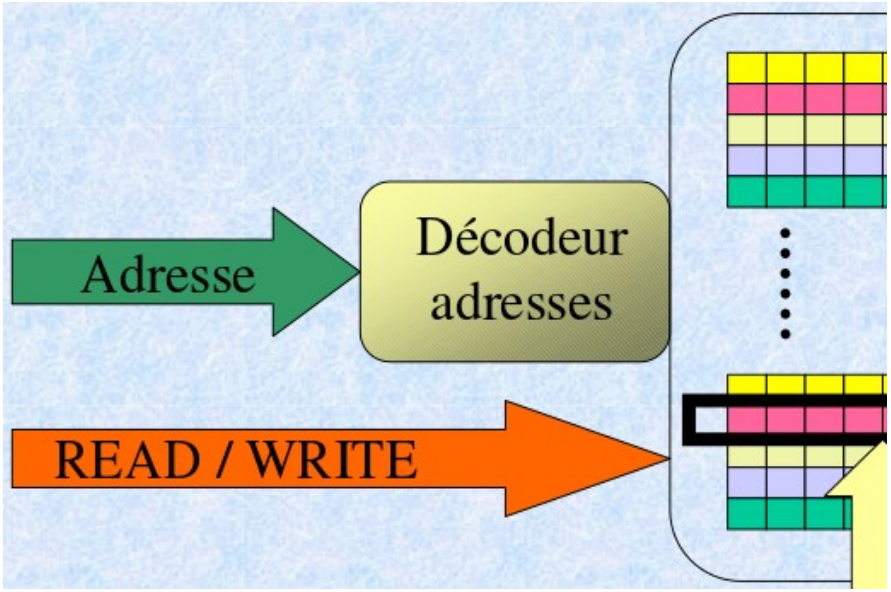
Mémoire


Memory representation
Read
0
1
2
Address bus
Decode
n
Memor
2n
2n-1

Mémoire

Sur la détection d'une adresse en entrée le décodeur sélectionne le mot mémoire

- Si signal Read alors restitution du mot mémoire en sortie
- Si signal Write alors modification du mot mémoire



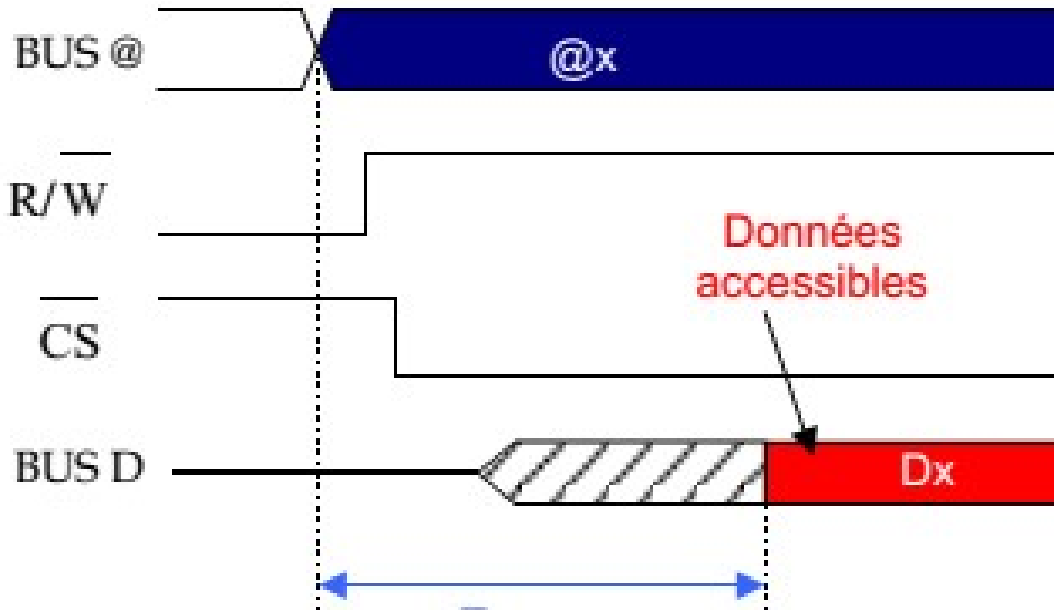
Processus lecture écriture dans la mémoire

Mémoire : Chronogramme de lecture/écriture

1. Le microprocesseur positionne son bus adresses sur la case mémoire qu'il veut lire ou écrire.
2. Les validations de boîtier CS i sont alors réalisées.
3. Le microprocesseur positionne alors sa broche lecture RD ou écriture WR pour indiquer à la mémoire le moment où elle doit mettre l'information sur le bus de données.
4. Le temps que met la mémoire pour le faire s'appelle le temps d'accès de la mémoire.
5. Tous les signaux sont séquencés par une horloge. Le microprocesseur n'agit que sur les fronts d'horloge.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

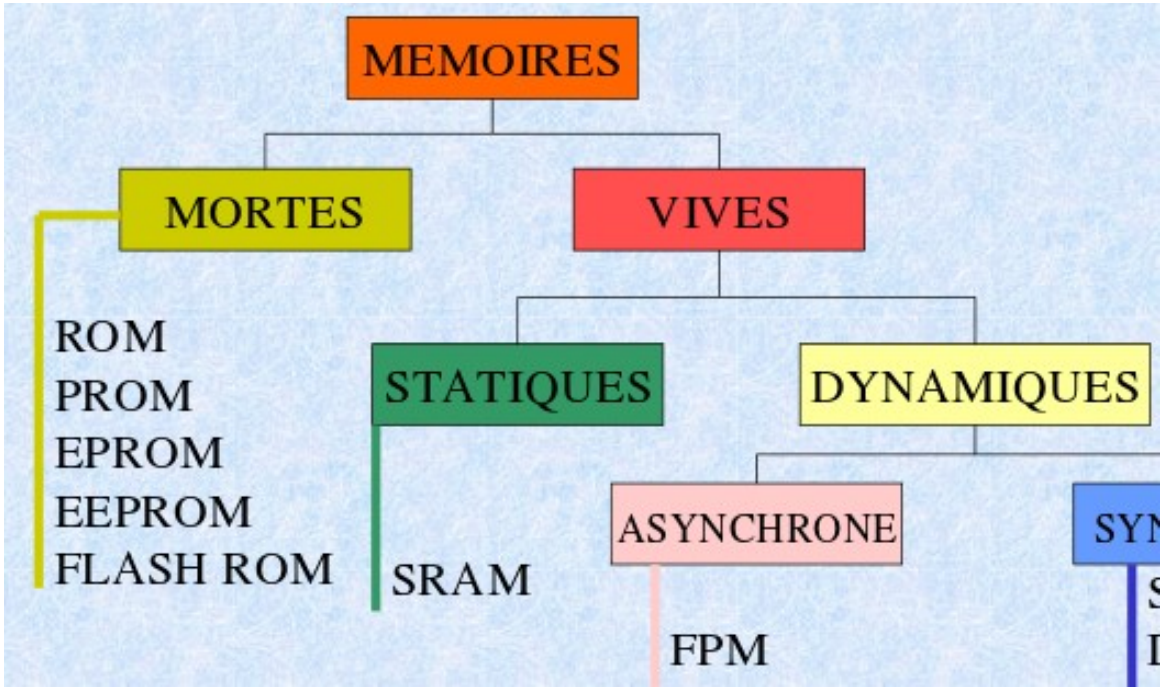
Mémoire : Chronogramme de lecture/écriture



Mémoire : Caractéristique de la mémoire

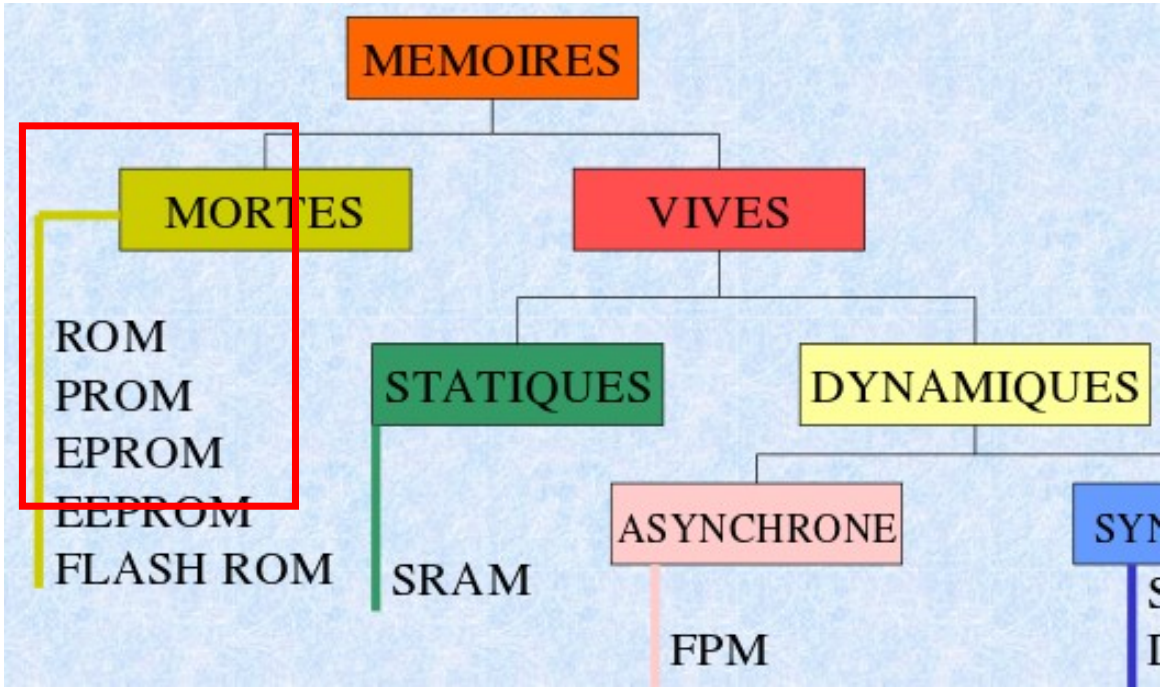
1. **Capacité** : Dite aussi taille de la mémoire, elle correspond au nombre d'informations qu'elle peut contenir. Elle est généralement exprimée en bits ou en nombre de mots mémoire.
2. **Temps d'accès** : C'est le temps qui sépare le début de l'opération d'accès et sa terminaison. Dans la pratique et pour plusieurs types de mémoire, le temps que demande une opération de lecture peut être différent de celui d'une opération d'écriture. Dans ce cas nous comptons le temps le plus long.
3. **Cycle mémoire** : C'est le temps minimal se coulant entre deux accès successifs à la mémoire. Théoriquement, le cycle mémoire est égal au temps d'accès.
4. **Débit** : C'est le nombre d'informations (exprimé en bits) lues ou écrites par seconde.
5. **Volatilité** : Elle caractérise la permanence des informations dans la mémoire centrale.

Type des mémoires : Classification des mémoires



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Type des mémoires : Mémoires Morts



Type des mémoires : Mémoires Morts

Les mémoires mortes (ROM) Pour certaines applications, il est nécessaire de pouvoir conserver des informations de façon permanente même lorsque l'alimentation électrique est interrompue. On utilise alors des mémoires mortes ou mémoires à lecture seule (ROM : Read Only Memory).

Ces mémoires sont non volatiles. Ces mémoires, contrairement aux RAM, ne peuvent être que lue. L'inscription en mémoire des données reste possible mais est appelée programmation. Suivant le type de ROM, la méthode de programmation changera. Il existe donc plusieurs types de ROM :

- ROM,
- PROM,
- EPROM,
- EEPROM,
- FLASH EPROM.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

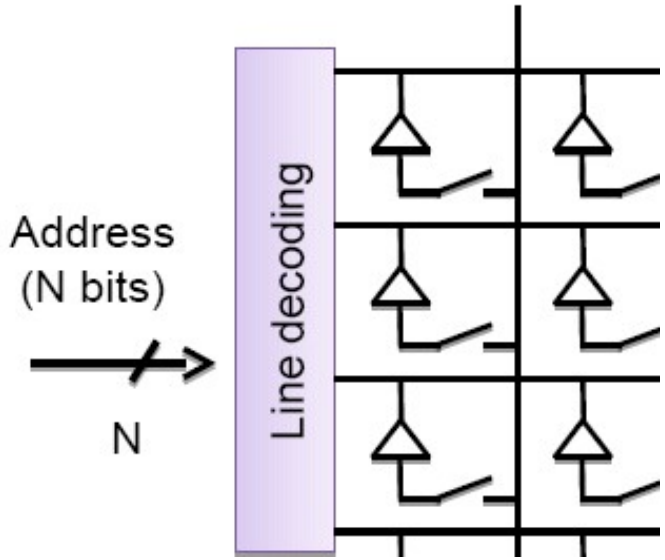
Type des mémoires : Mémoires Morts ROM (Read Only Memory)

ROM est un circuit intégré dont le contenu est déterminé une fois pour toute au moment de la fabrication.

Le coût relativement élevé de leur fabrication impose une fabrication en grandes séries, ce qui complique la mise à jour de leur contenu.

Au départ, ces mémoires étaient utilisées pour stocker les parties bas-niveau du système d'exploitation de l'ordinateur (BIOS du PC par exemple).

are connected through diodes.



Type des mémoires : Mémoires Morts ROM (Read Only Memory)

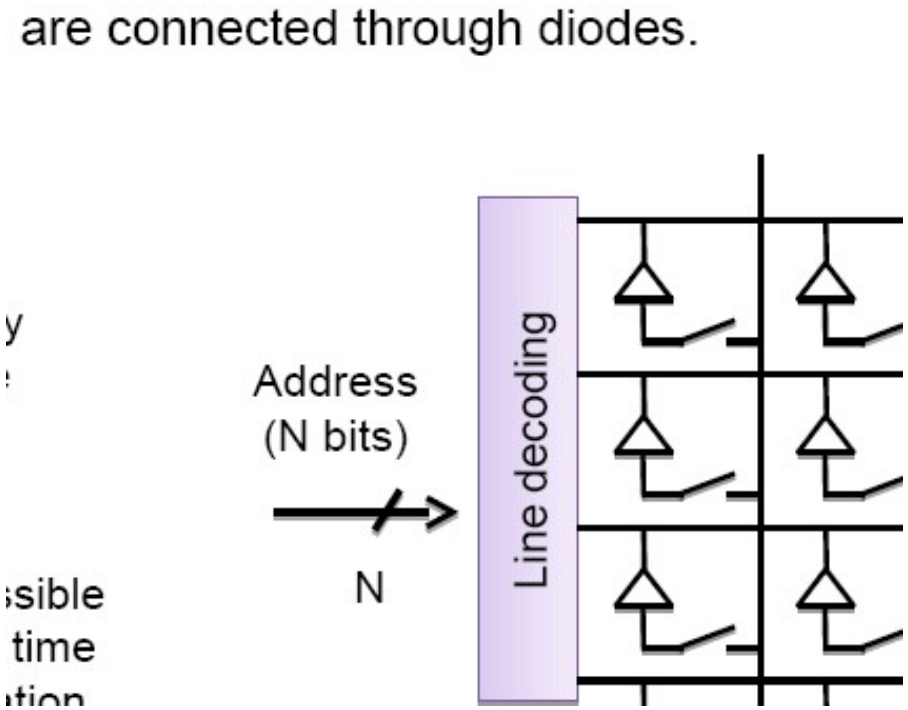
Les lignes sont connectées par des diodes.

Avantages:

- Haute densité
- Non volatil
- Vite

Désavantages:

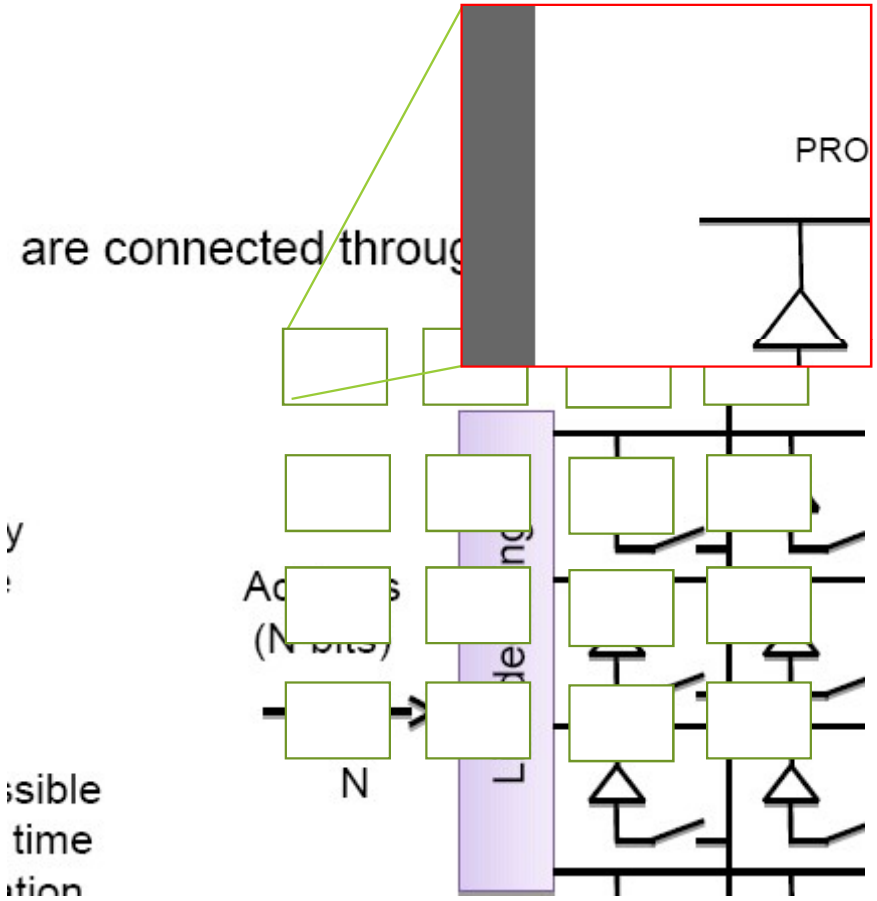
- Écrire impossible
- Temps de fabrication
- Aucune modification
- Coût élevé



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Type des mémoires : Mémoires PROM (Programmable ROM)

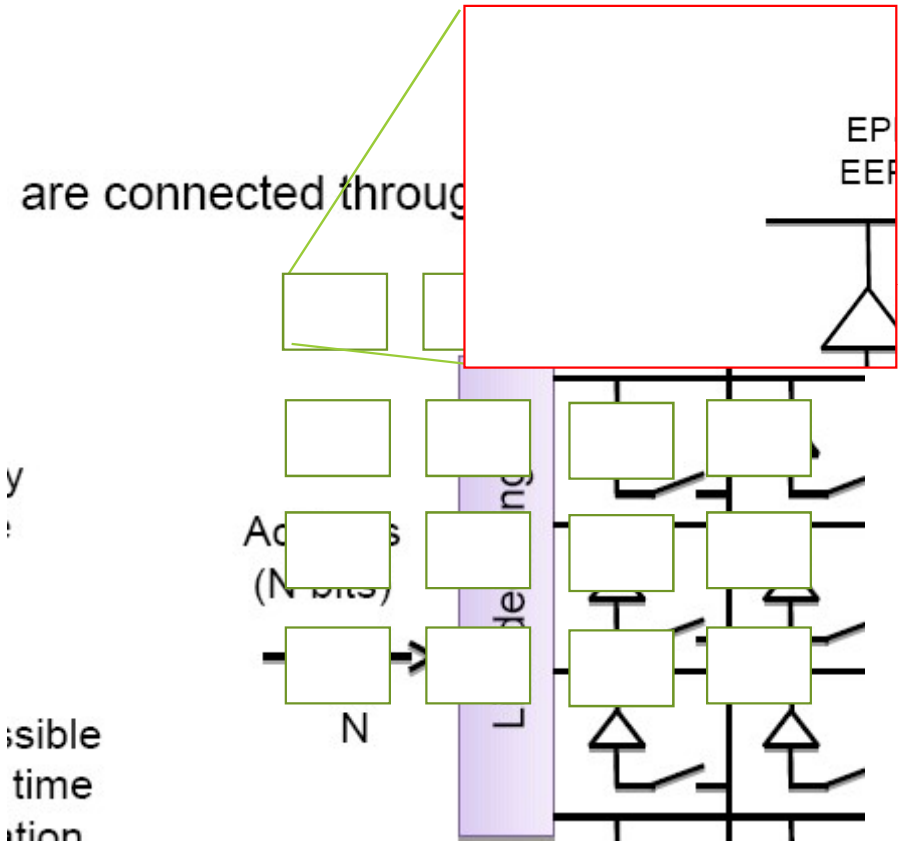
PROM (Programmable ROM) Alors que la mémoire ROM est enregistrée de manière irréversible lors de sa fabrication, la mémoire PROM est configurée par l'utilisateur en utilisant un programmateur de PROM, utilisé pour enregistrer son contenu. Le circuit PROM ne peut plus être modifié par la suite.



Type des mémoires : Mémoires EPROM (Erasable PROM) && EEPROM (Electricaly Erasable PROM)

Les mémoires EPROM sont des PROM reconfigurables : il est possible de les effacer pour les reprogrammer. L'effaçage se produit en exposant le boitier à un ***rayonnement ultraviolet (UV)***.

EEPROM (Electricaly Erasable PROM) Même principe qu'une EPROM, mais l'effacement se fait à ***l'aide de signaux électriques***, ce qui est plus rapide et pratique.

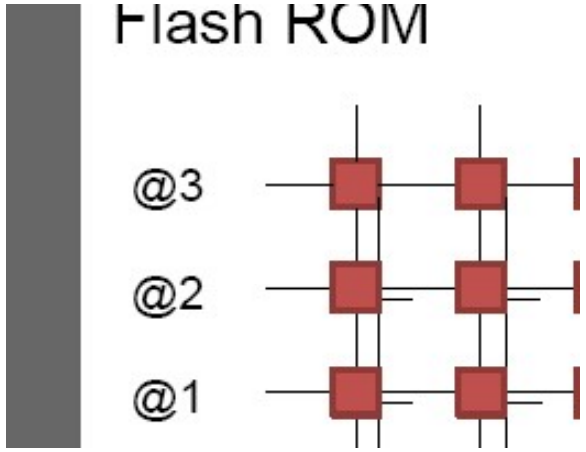


Type des mémoires :Mémoires FLASH EPROM (Flash disque et carte mémoire)

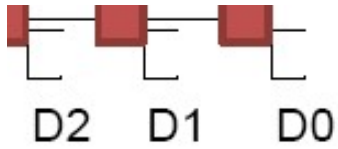
FLASH EPROM (Flash disque et carte mémoire) Les mémoires FLASH sont similaires aux mémoires EEPROM, mais l'effacement peut se faire par sélectivement par blocs et ne nécessite pas le démontage du circuit.

La Flash EPROM a connu un essor très important ces dernières années avec le boom de la téléphonie portable et des appareils multimédia (PDA, appareil photo numérique, lecteur MP3, etc...).

EEPROM NOR: Read/write by byte

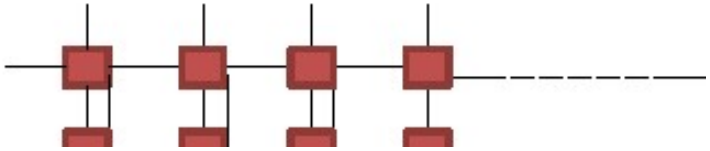


Flash EEPROM NAND: Read/write by sector

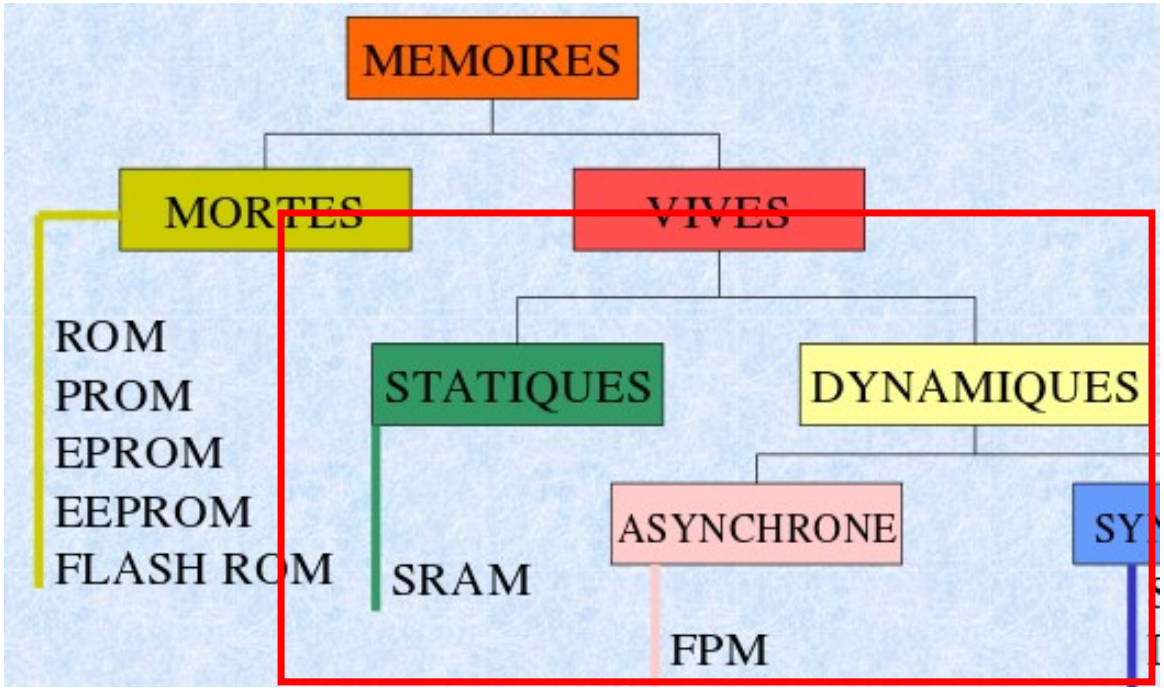


AND:

Secteur 1= 0 à n



Type des mémoires : Mémoires Vives



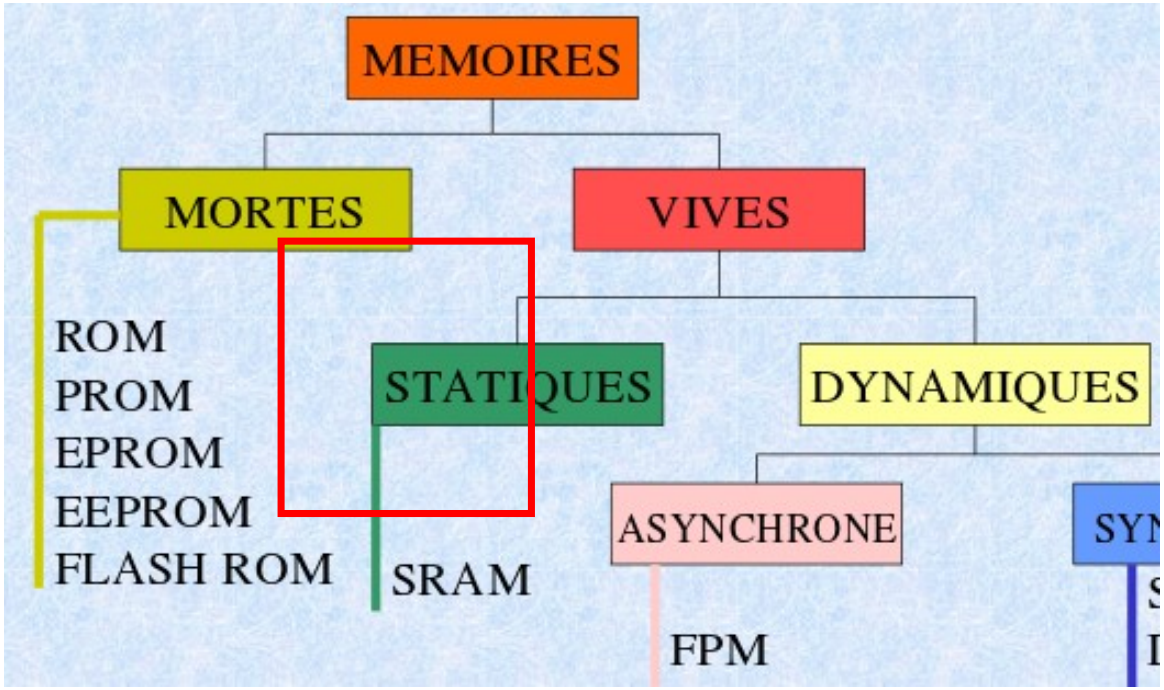
Type des mémoires : : Mémoires Vives

Mémoires vives (RAM) Une mémoire vive sert au stockage temporaire de données. Elle doit avoir un temps de lecture et écriture très court pour ne pas ralentir le microprocesseur. Les mémoires vives sont en général volatiles : elles perdent leurs informations en cas de coupure d'alimentation.

Certaines d'entre elles, ayant une faible consommation, peuvent être rendues non volatiles par l'adjonction d'une batterie. Il existe deux grandes familles de mémoires RAM (Random Acces Memory : mémoire à accès aléatoire) :

- Les RAM statiques,
- Les RAM dynamiques,

Type des mémoires : Mémoires Vives

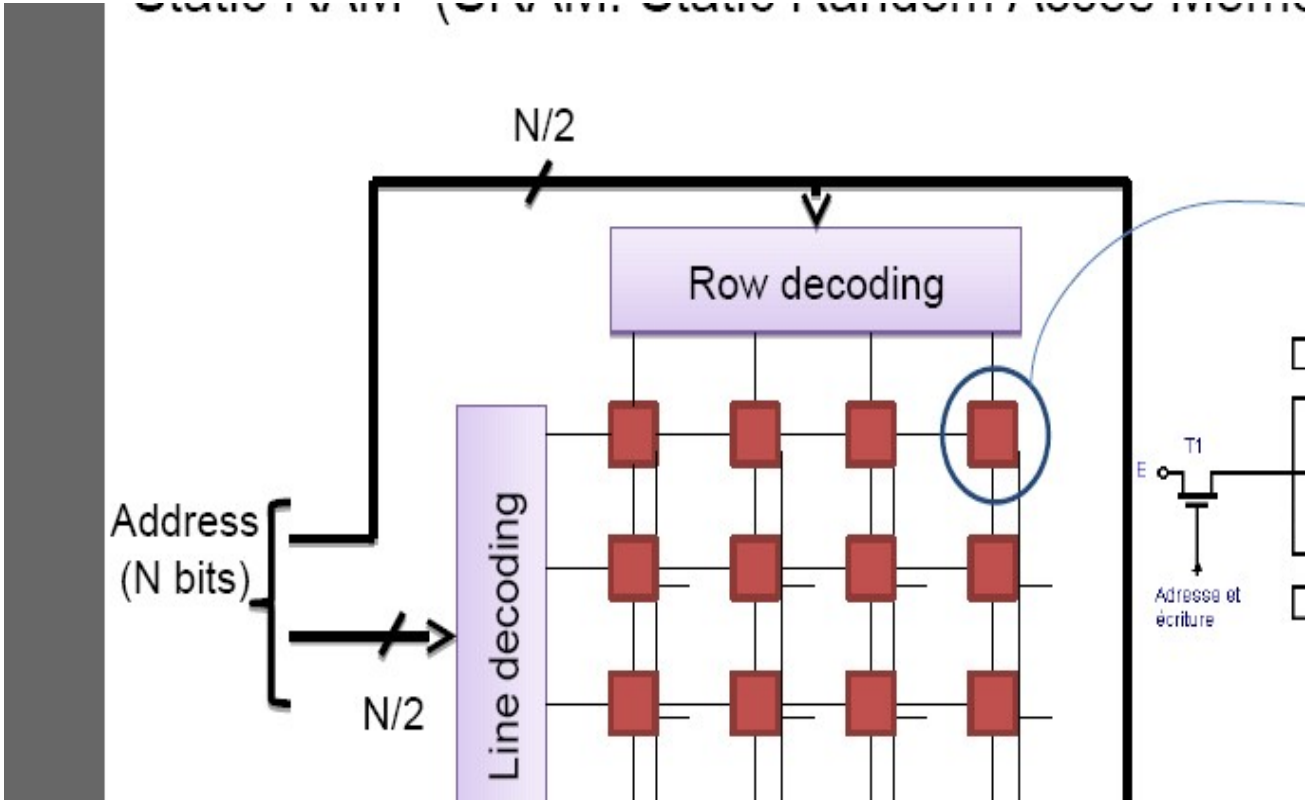


Mr. MEGNAFI Hicham (ESSA-Tlemcen)

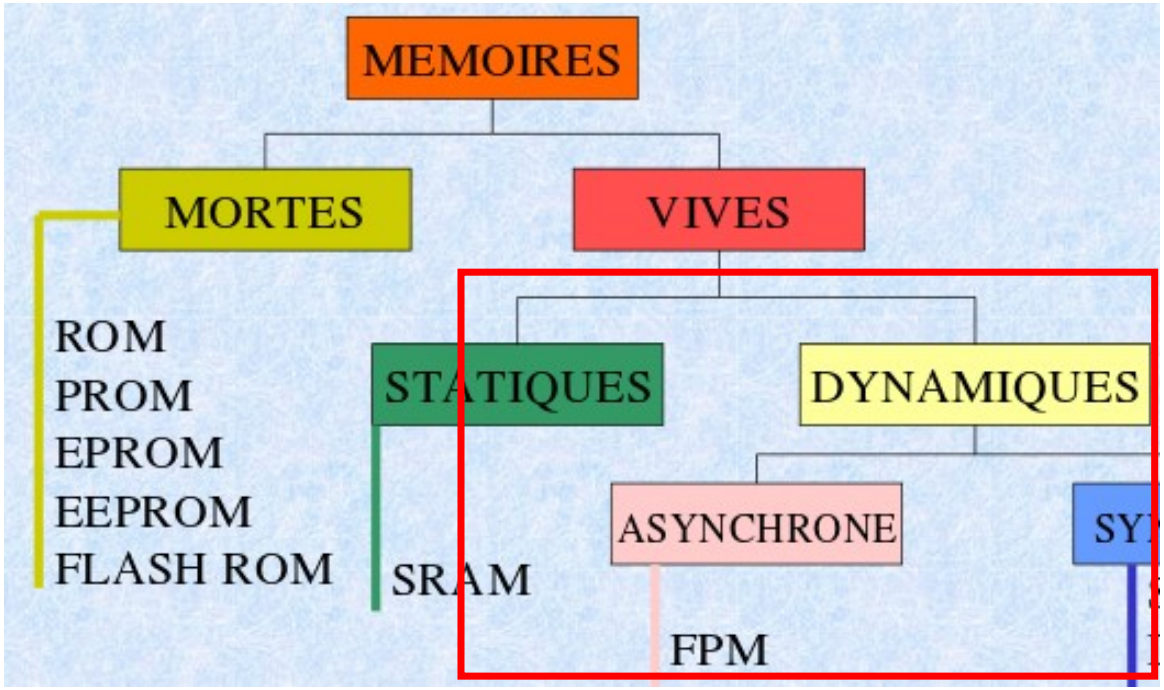
Type des mémoires : Static RAM (SRAM: Static Random Acces Memory)

RAM statiques Le bit mémoire d'une RAM statique (SRAM) est composé d'une bascule. Chaque bascule contient entre 4 et 6 transistors.

Les SRAM permettent des temps d'accès court à l'information.



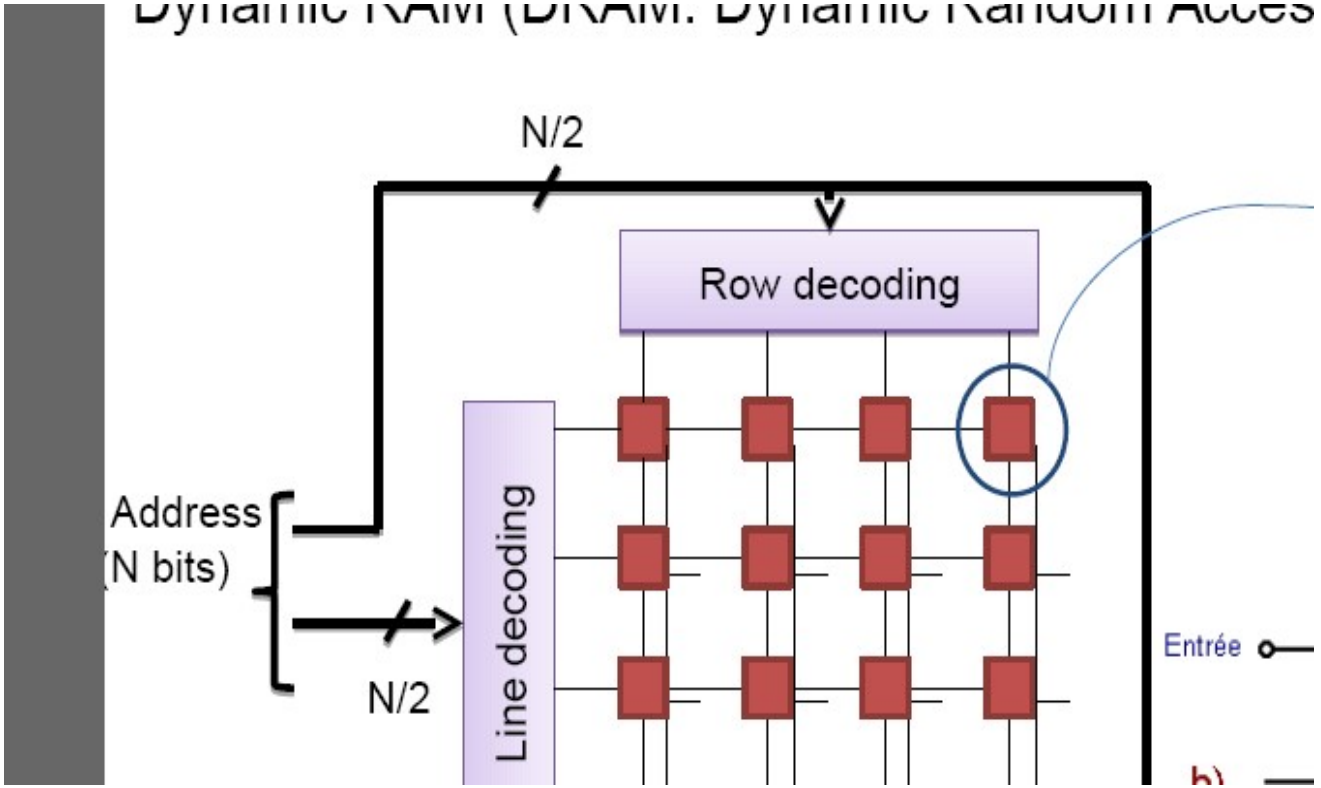
Type des mémoires : Mémoires Vives



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Type des mémoires : Dynamic RAM (DRAM: Dynamic Random Acces Memory)

Les RAM dynamiques Dans les RAM dynamiques (DRAM), chaque bit est
réalisé à partir d'un transistor relié à un petit condensateur.
L'état chargé ou déchargé du condensateur permet de distinguer deux états
(bit 0 ou bit 1).

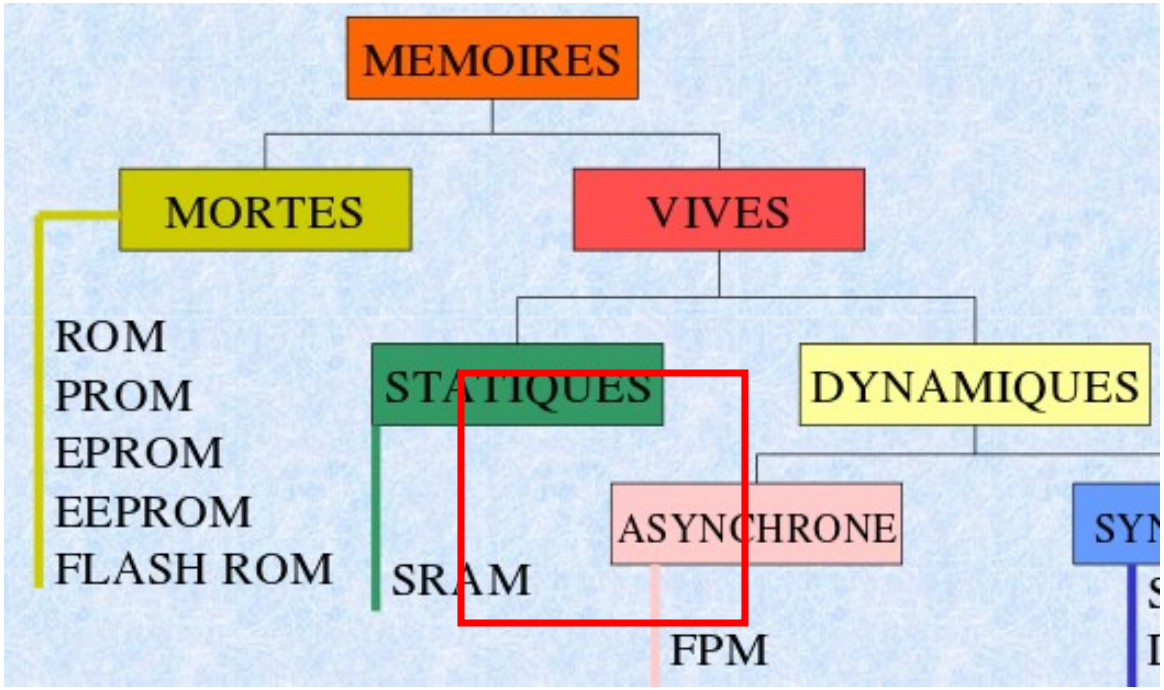


Type des mémoires : Mémoire vive dynamique asynchrone & synchrone

Mémoire asynchrone : le processeur ne peut pas engager un nouvel accès tant que l'accès précédent n'est pas achevé,

Mémoire synchrone : le processeur peut engager des accès consécutifs même si l'accès précédent n'est pas achevé.

Type des mémoires : Mémoires Vives

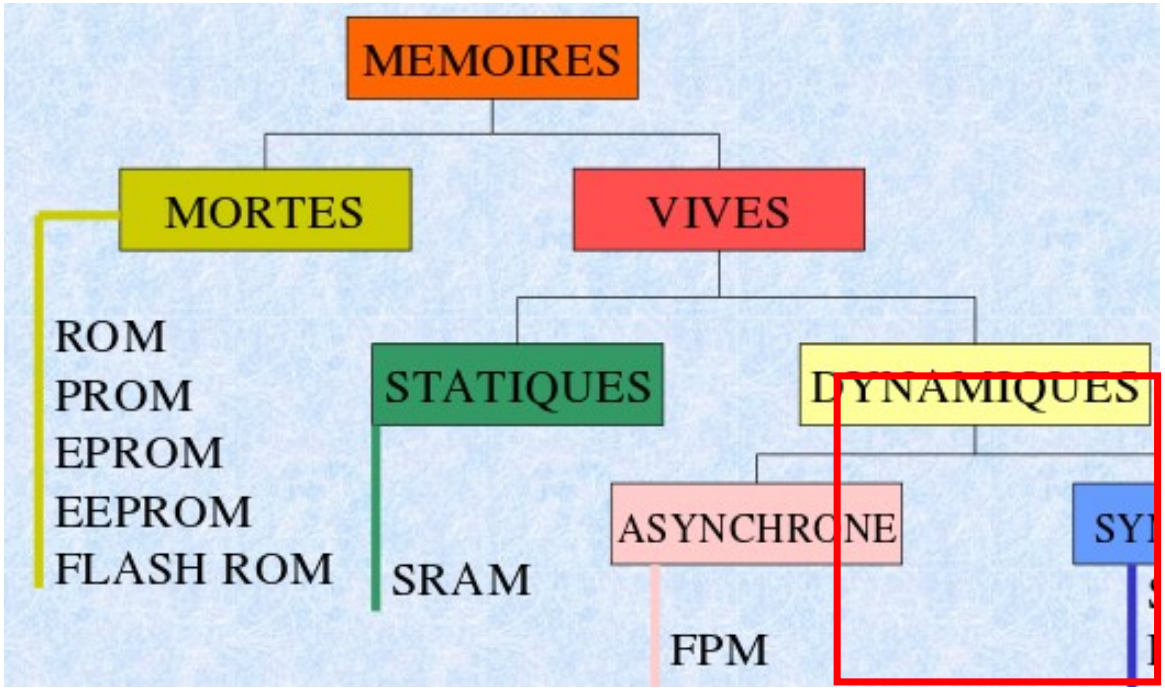


Type des mémoires : Mémoire vive dynamique asynchrone

Mémoire DRAM asynchrone

- FPM (Fast Page Mode), est une mémoire Dram standard qui est disponible avec une vitesse de 60ns.
- EDO (Extended Data Output), amélioration du standard FPM.

Type des mémoires : Mémoires Vives

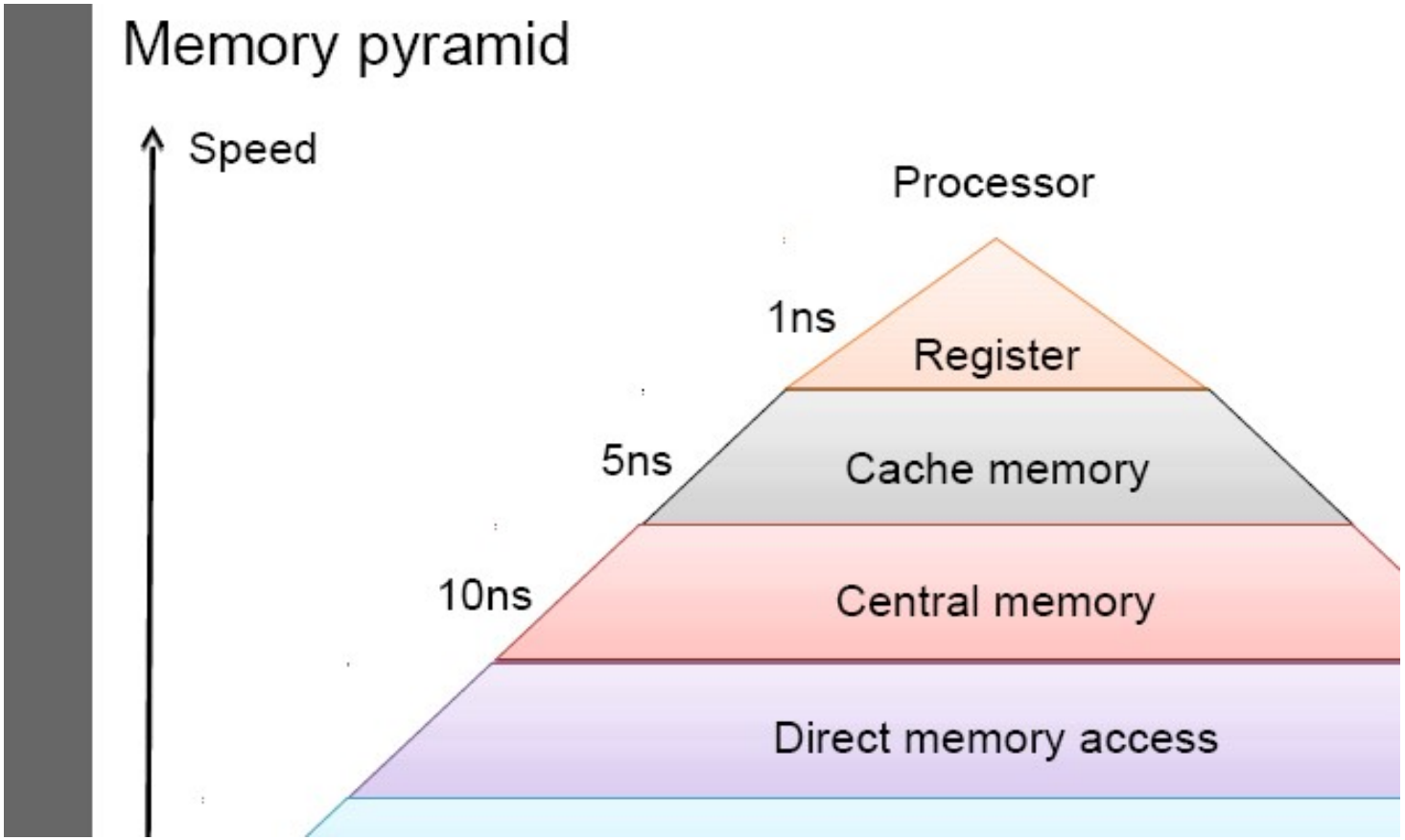


Type des mémoires : Mémoire vive dynamique synchrone

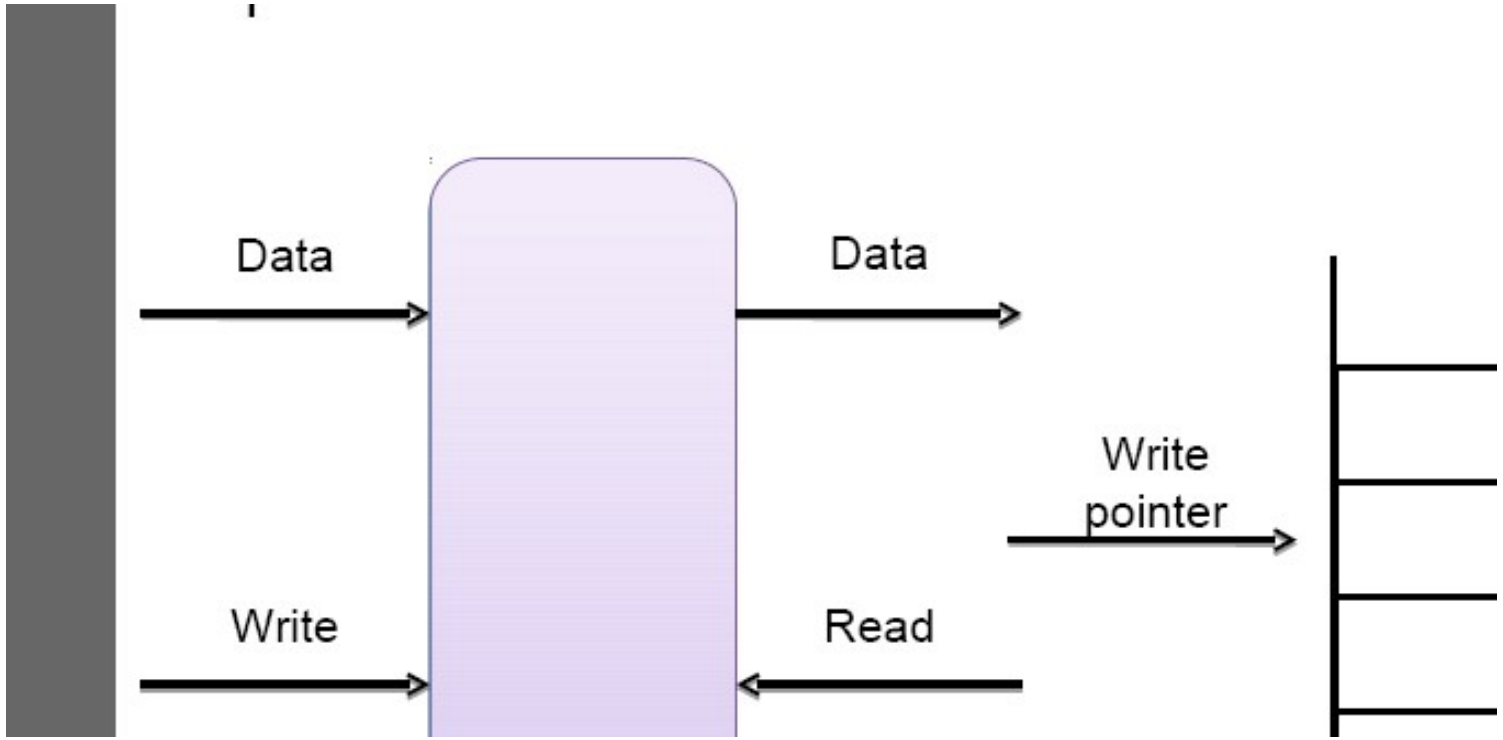
SDRAM(Synchronous DRAM) : en très forte évolution, échange ses données avec le processeur en se synchronisant avec ce dernier ce qui permet d'éviter les états d'attente, cette technologie de mémoire est en train de connaître une importante progression du fait des temps d'accès autorisés de l'ordre de 10 à 12 ns, allant de paire avec une baisse non négligeable des coûts.

DDR SDRAM : Samsung, Nec et Toshiba se sont associés pour mettre au point un nouveau standard dit DDR (Double Data Rate) qui doublerait le taux de transfert des actuelles SDRAM. La SDRAM deviendrait donc DDR SDRAM ou SDRAM II et offrirait des taux de transfert atteignant 1,03 GO/s. elle est en effet capable de lire les données aussi bien sur le front montant que sur le front descendant du signal d'horloge. Ce qui permet de doubler son taux de transfert.

Pyramide de la mémoire

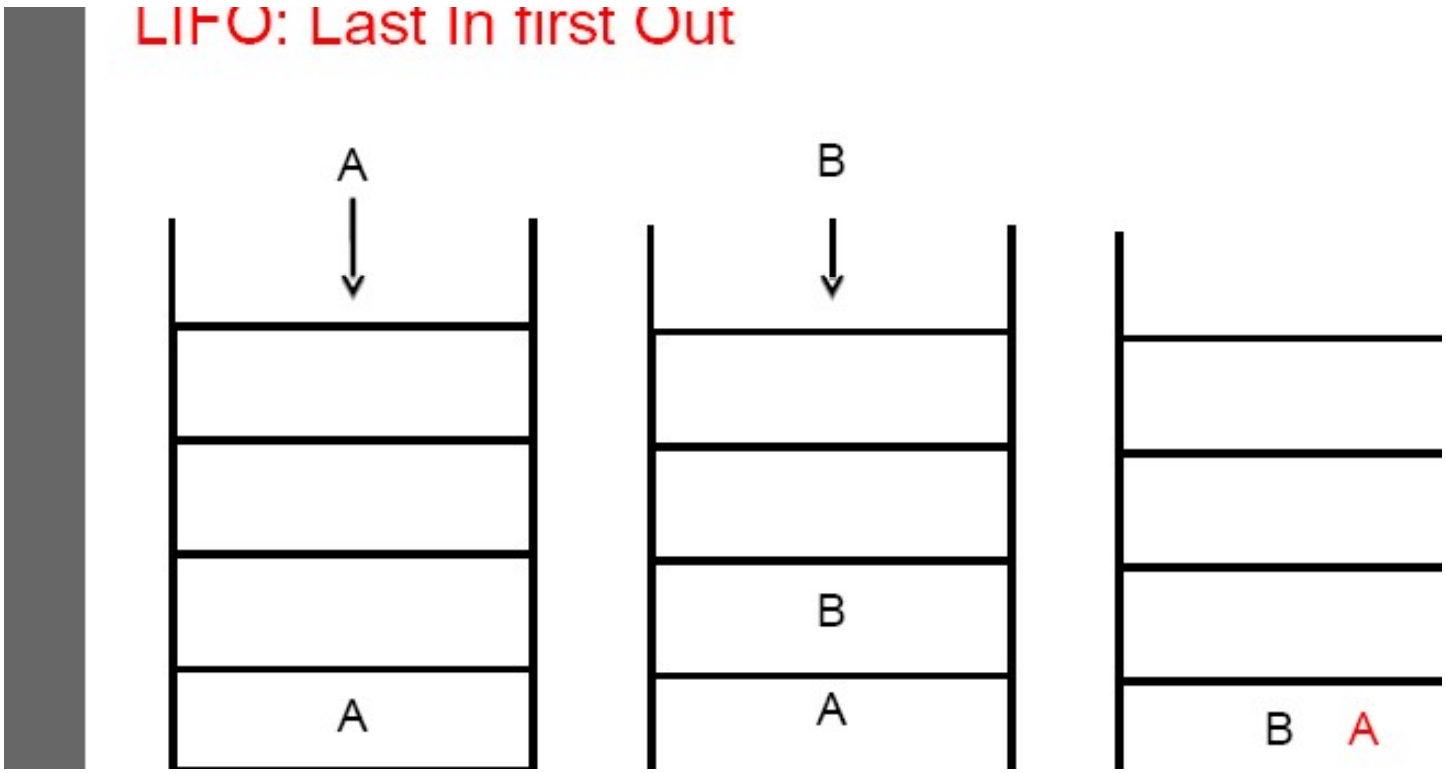


Mémoire séquentielle



FIFO: First In First Out

LIFO: Last In first Out



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

1 bit = information de base = *0 ou 1*

1 octet (byte) = 8 bits

1 0 1 0 0 1 0 1 = 165 $_{10}$

Une mémoire est caractérisée par :

- Sa capacité exprimée en : Bit, Kbit , Mbit, Gbit ou octets, Ko, Mo, Go
- Son temps d 'accès : Exprimé en ns, il renseigne sur la vitesse de la mémoire
- Le format de son bus de donnée 1bit, 4bits, 8bits...

1 Kilo = 2^{10} = *1.024* **1Méga = 2^{20} = *1.048.576*** **1Giga = 2^{30} = *1.073.741.824***

TD

Question 1

Le microcontrôleur 16F84 implémente une mémoire programme ?

- ❑ dont l'adresse est comprise entre 0x000 et 0x3FF
- ❑ dont l'adresse est comprise entre 0x000 et 0x3FFF
- ❑ dont l'adresse est comprise entre 0x000 et 0x3FFFF

Question 2

Un microcontrôleur PIC exécute 5 millions d'instructions par seconde:

- ❑possède un cycle instruction de 100 nano sec ;
- ❑possède un cycle instruction de 200 nano sec ;
- ❑possède un cycle instruction de 500 nano sec ;

Question 3

Dans un programme, le PIC 16F84:

- ❑A la possibilité d'exécuter 5 appels consécutifs imbriqués de sous-programmes;
- ❑A la possibilité d'exécuter 8 appels consécutifs imbriqués de sous-programmes;
- ❑A la possibilité d'exécuter 14 appels consécutifs imbriqués de sous-programmes;

Question 4

La mémoire vive (RAM) est :

- ❑Une unité de stockage permanente d'informations.
- ❑Une unité de stockage volatile d'informations.
- ❑Un système de sécurité des ordinateurs.

Question 5

Cochez la case correspondant à la mesure la plus grande ?

- ❑ 1 Ko
- ❑ 1 Gb
- ❑ 2 Mb
- ❑ 1200 Ko

Question 7

Un méga-octet (Mo) vaut :

- ❑1000 Ko
- ❑1 048 576 octets
- ❑1 000 000 octets
- ❑1024 octets

Question 8

Quelle affirmation est fausse ?

- ❑La RAM stocke des informations de manière temporaire
- ❑La ROM peut être lue, mais son contenu ne peut être effacé
- ❑La mémoire cache est un CD-Rom

Question 9

L'accès à la mémoire RAM est plus rapide que l'accès au disque dur?

- ❑Oui
- ❑Non
- ❑Non sauf si la capacité de la RAM est inférieure à 256 Mo
- ❑oui à condition que le disque dur sera d'une marque dépassée

Question 10

Lequel de ces types de memoire est le plus rapide ?

- ❑Registre
- ❑disque optique
- ❑Cache
- ❑memoire principale (ram)

Question 11

Pour stocker un fichier de 337 920 octets et le transporter chez un ami, je choisis comme support :

- ❑ Un disque dur de 300 Go
- ❑ Un cédérom de 700 Mo
- ❑ Un dévédérom de 4,7 Go
- ❑ Une clé USB de 2 Go

Question 12

Quel est la différence entre une mémoire vive statique et dynamique ?

Question 13

Quels sont les caractéristiques de la mémoire ?

Question 14

Type des mémoires ROM sont :

Question 15

Différences entre Mémoires vive dynamique asynchrone et synchrone :

Plan de Cours

1. Projets réalisés
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. ***Les Bus de communication***
7. Les Entrés/ Sorties
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

6.1 **Les Bus de communication**

6.2 **Principe de communication**

- ❑ **Lecture & Écriture**

6.3 Types de bus

- ❑ **Interne : SATA, PCI, AGB**
- ❑ **BUS externes: USB, *I2C***

Les Bus de communication

Définition:

Un bus est un mécanisme qui permet le transfert de données entre éléments dans l'ordinateur(bus mémoire : composé d'un bus d'adresse et d'un bus de données).

- ❑ Ensemble de fils chargés de relier les différents éléments de l'ordinateur.
- ❑ Caractérisé par son nombre de fils, la nature des informations véhiculées, le mode de fonctionnement (série ou parallèle, synchrone ou asynchrone).

Principe de communication

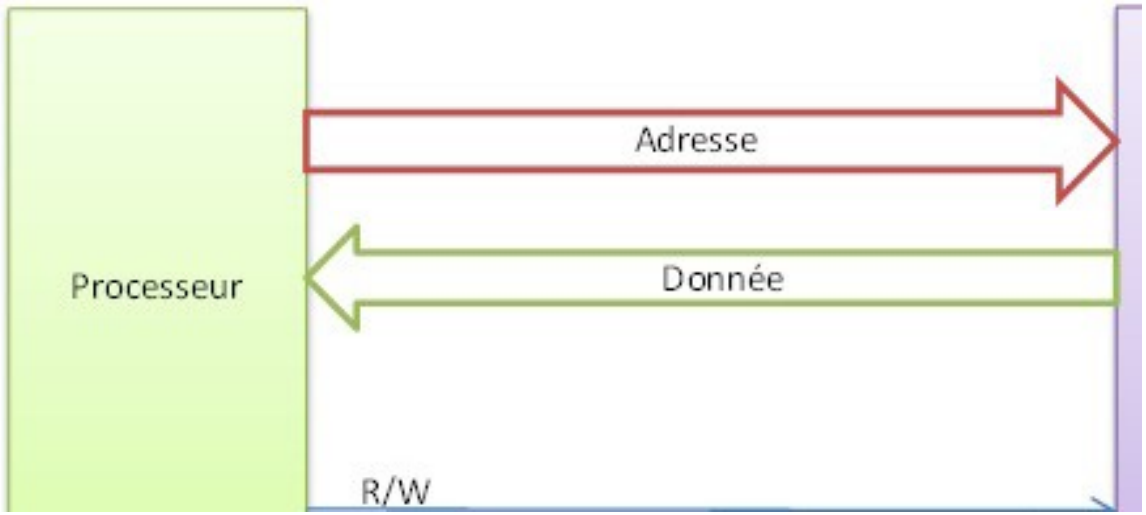


Figure : Principe de communication processeur périphérique

Principe de communication Lecture

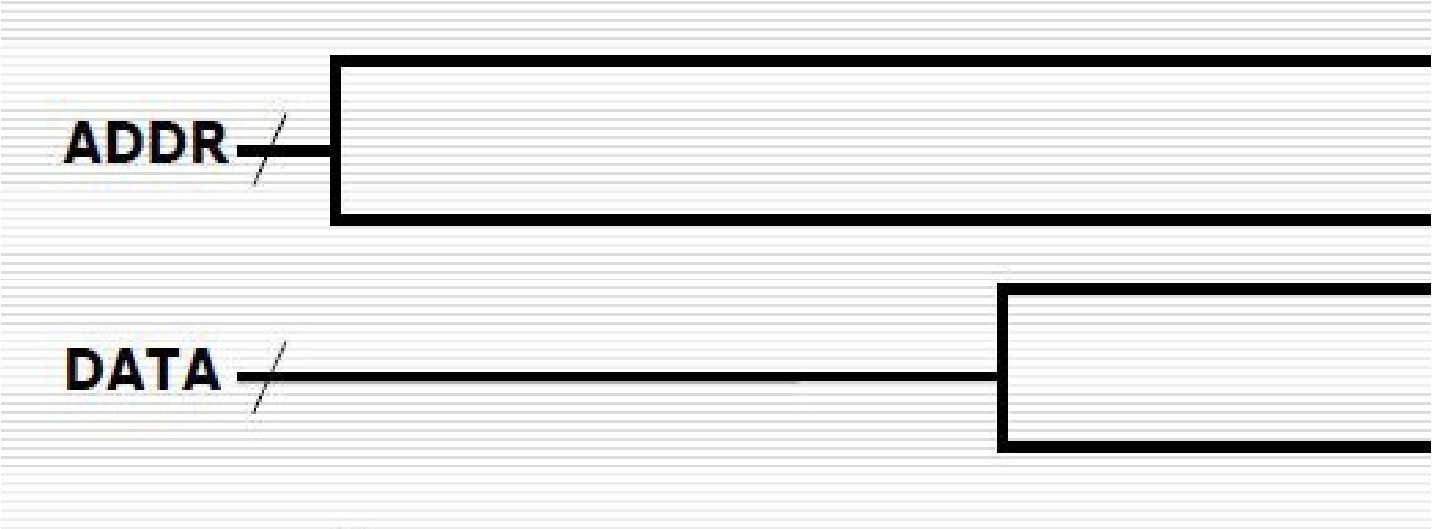


Figure : Principe de lecture

Principe de communication Écriture

Figure : Principe d'Écriture

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Principe de communication lecture

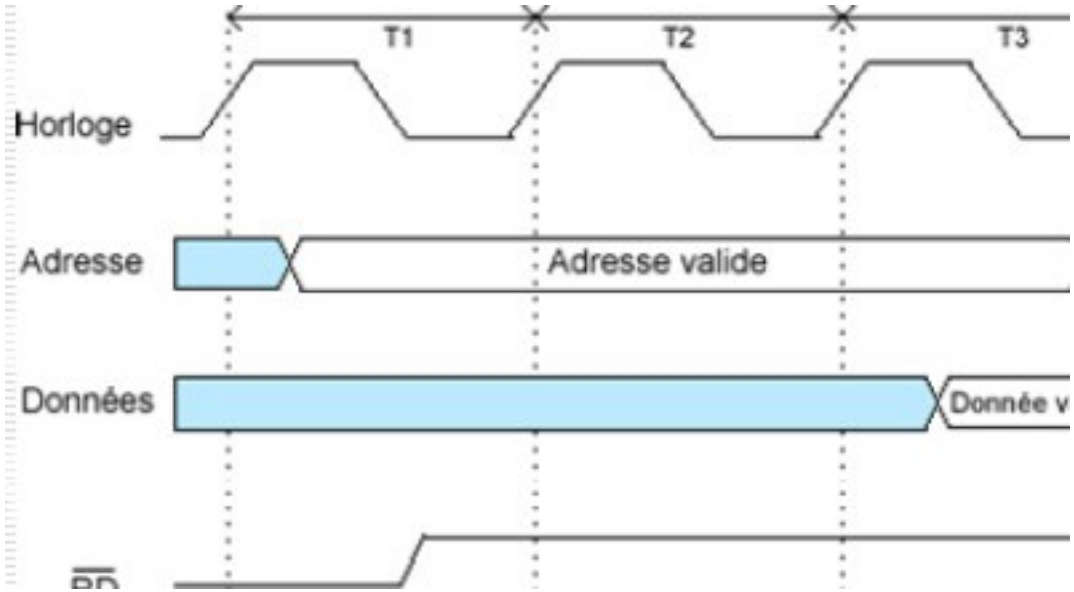


Figure : Chronogramme de lecture

Types de bus

Les différents types de bus

- **Bus interne :** connecte les composants internes de l'ordinateur (CPU, mémoire, graphique. . .)
- **Bus externe :** relie les périphériques externes
- **Bus série** : les données sont transmises les une à la suite des autres.
- **Bus parallèle :** plusieurs données sont transmises en parallèle

Bus interne :

Les bus internes, également connus sous le nom de bus de données ou bus mémoire, relient les composants internes principaux d'un ordinateur, tels que le processeur et la mémoire, sur la carte mère. Le bus de données interne est également considéré comme un bus local, parce qu'il est destiné à se connecter à des périphériques locaux. Ce bus est généralement assez rapide et est indépendant du reste des opérations informatiques.

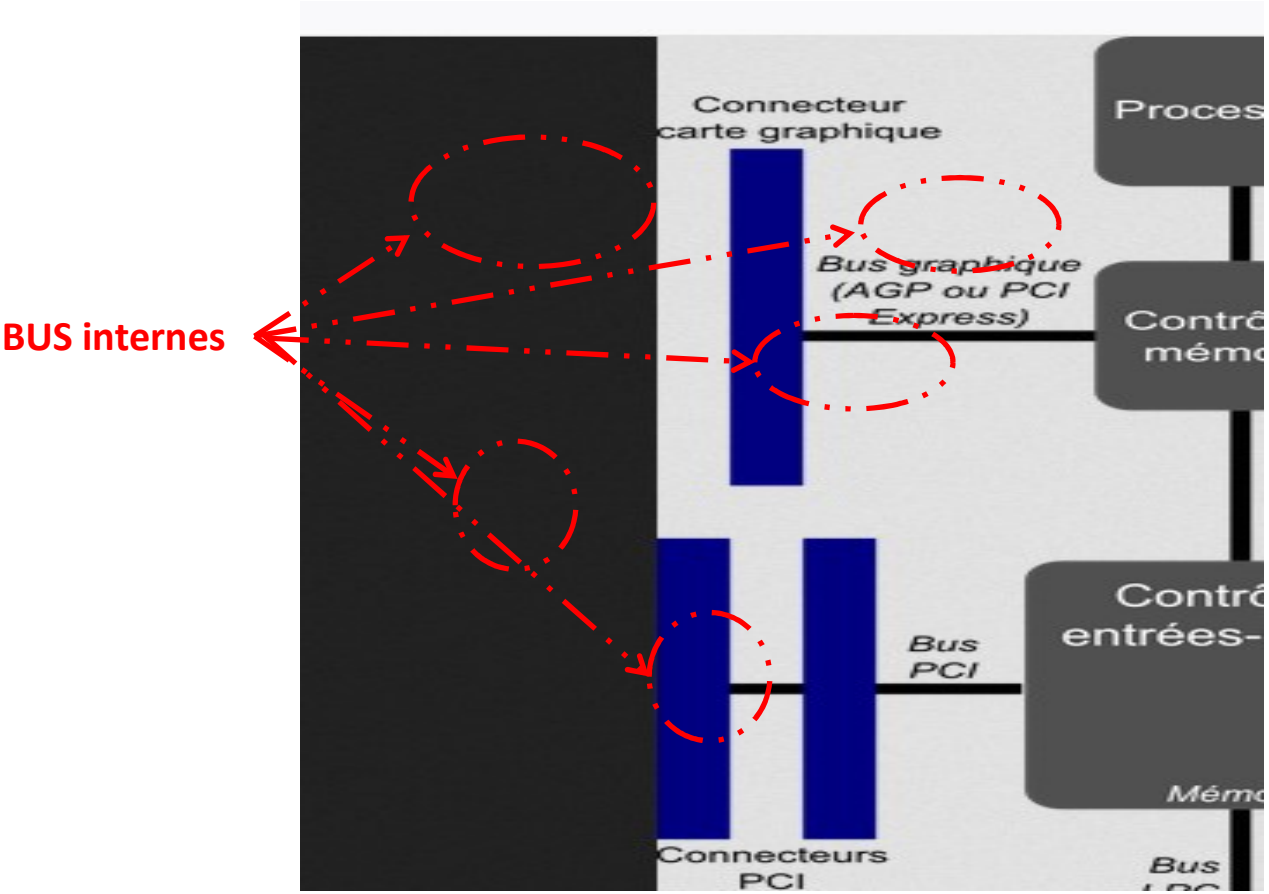
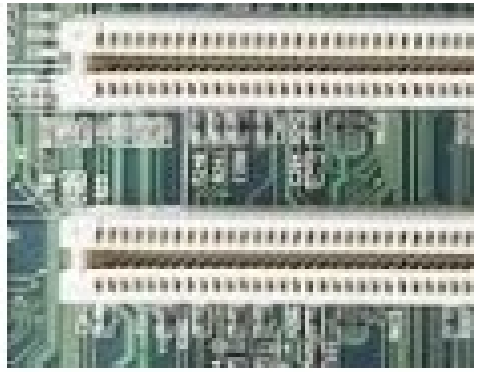


Schéma de la relation entre le processeur et les bus

Types de bus : BUS internes

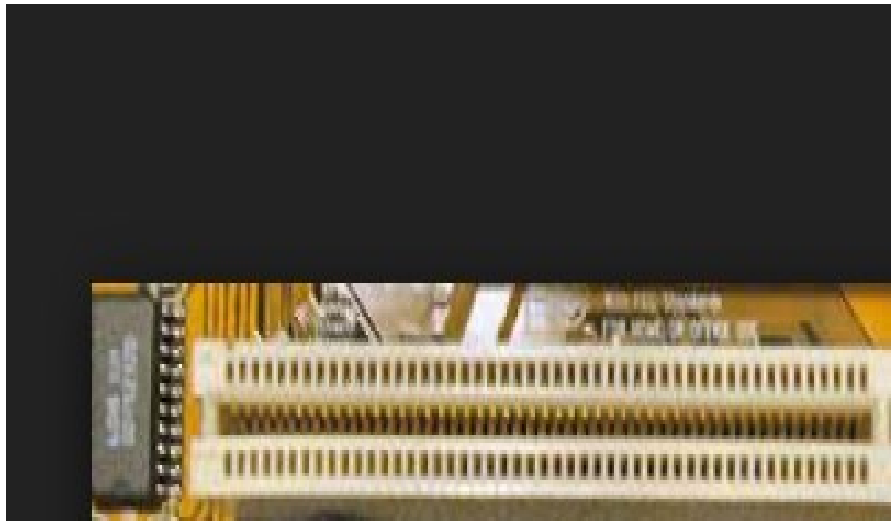
Les bus d'extensions : PCI

- PCI (1992) : Peripheral Componement Interconnect)
- Existe en version 32 et 64 bits.
- PCI 2.3 qui existe en deux versions :
  - Bus 32 bits à 33 MHz (soit une bande passante maxi de 133 Mo/s) (la plus répandue)
  - Bus 64 bits à 66 MHz (soit une bande passante maxi de 528 Mo/s), utilisé sur certaines cartes mères professionnelles ou sur des serveurs (elles font deux fois la longueur du PCI 2.2 à bus 32 bits) ;
- PCI 3.0 : 8 GT/s, PCI 4.0 : 16 GT/s, PCI 5.0 : 32 GT/s.

Types de bus : BUS internes

Les bus d'extensions : PCI

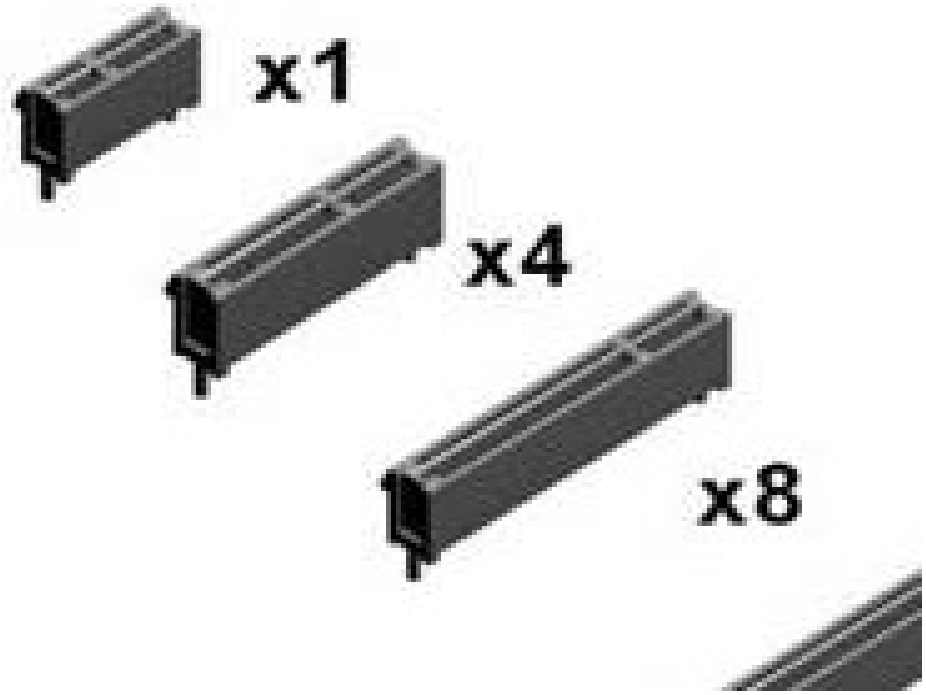


- Technique de transmission : Parallèle.
- Auto-configurable : les cartes connectées sont automatiquement détectées et configurées
- Dans sa version purement PCI la bande passante est partagée entre tous les éléments connectés sur le bus, contrairement à ce qui se passe pour la version PCI Express où elle est dédiée pour chaque périphérique. Cette dernière est donc préférable si on veut utiliser simultanément des cartes haut débit (carte réseau gigabits, contrôleur de disque, carte graphique…).

Types de bus : BUS internes

Types de bus : PCI

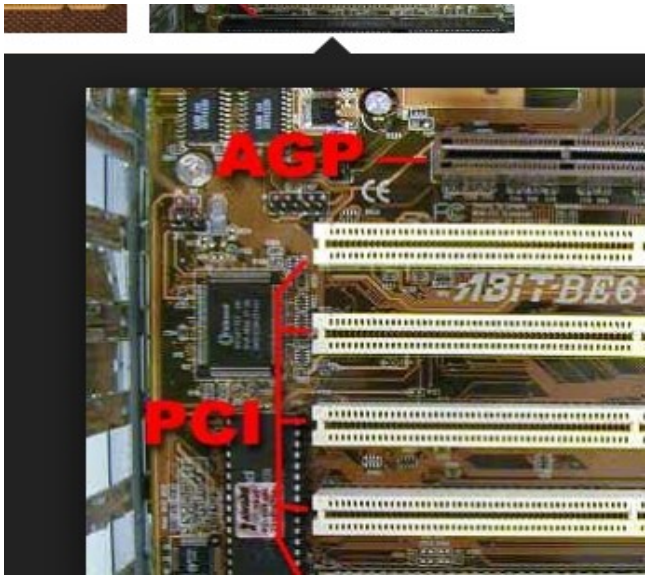
Le PCI-Express est une version plus petite et bien plus rapide du PCI grâce au passage de la technique de transmission parallèle à la technique de transmission série.



Les différentes versions du bus PCI-Express

Types de bus : BUS internes

Les bus d'extensions : AGP



- AGP (1997) : Accelerated Graphic Port. Bus dédié aux cartes graphiques.
- Le contrôleur graphique AGP utilise un bus 32 bits dédié à hautes performances qui lui offre un accès direct à la mémoire.
- Débit : 528Mo/s en AGP2x

- Le bus AGP donne également accès à la mémoire centrale, via le contrôleur de mémoire, ce qui permet d'utiliser celle-ci pour stocker des données graphiques supplémentaires (coordonnées), ainsi, il devient inutile d'acquérir de la mémoire vidéo supplémentaire pour bénéficier pleinement des fonctions 3D d'un circuit vidéo.

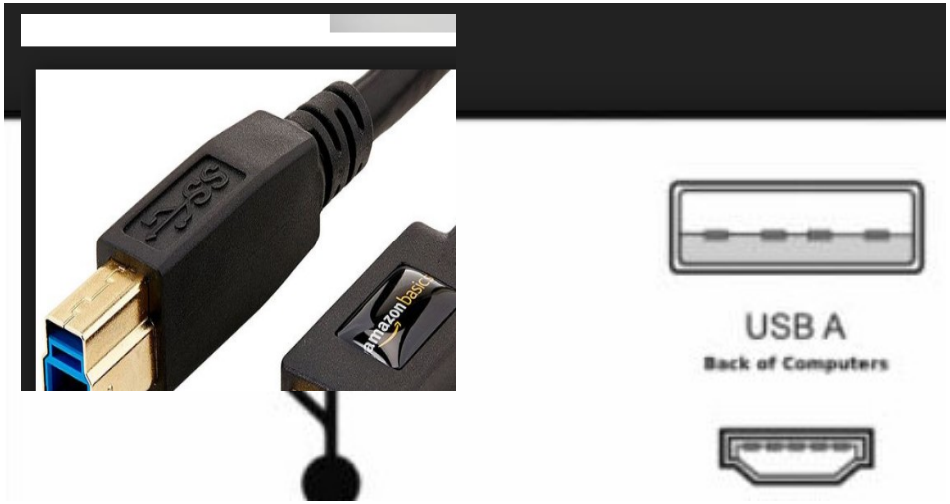
Types de bus : BUS externes

BUS externes

Le BUS externe, aussi appelé BUS d'extension ou d'entrée/sortie, relie le microprocesseur aux périphériques d'entrée et de sortie, tels que l'écran, le clavier, la souris, etc

Le Bus USB

L'Universal Serial BUS, autrement dit l'USB, est un BUS informatique dit « à transmission série », servant à connecter des périphériques informatiques à un ordinateur. Il y a différentes normes d'USB :



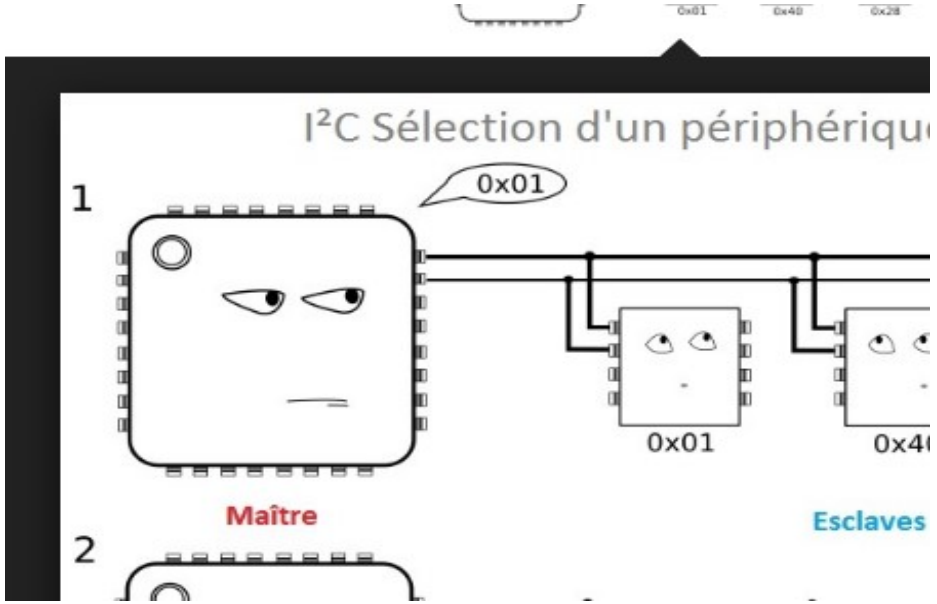
- ❑ L'USB 1.0
- ❑ L'USB 2.0 : majoritairement utilisé aujourd'hui
- ❑ L'USB 3.0 ou USB 3.0G : propose pour principale nouveauté un débit supérieur : 5 Gigabits/seconde. Alors que les 480 Mbps de l'USB 2.0 commençaient à se montrer, notamment pour les derniers disques durs ou clés USB, le passage à l'USB 3.0 devrait donner un coup de fouet non négligeable aux taux de transfert.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Types de bus : BUS I2C

Le bus I2C

I2C (Inter Integrated Circuit Bus)est un bus série permettant de transmettre des informations de façon asynchrone entre divers circuits connectés sur le bus.



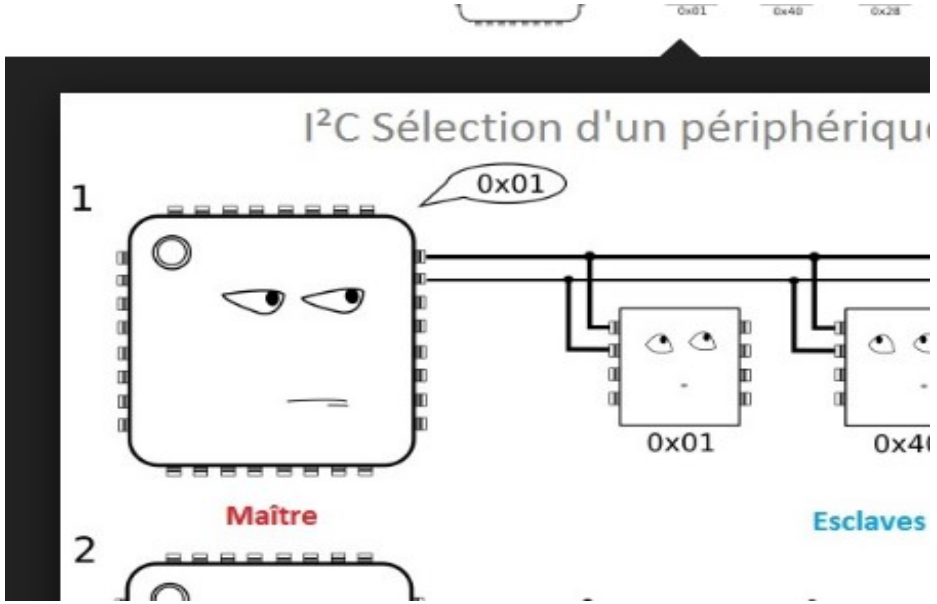
- Le protocole de la liaison est du type MAITRE/ESCLAVE.
- Chaque circuit (ordinateur, microcontroleur,ecran,...) est reconnu par son adresse.
- peut être soit transmetteur soit receveur de l'information

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Types de bus : BUS I2C

Terminologie du bus I2C

- **Transmetteur** : Le circuit qui envoie la donnée sur le bus
- **Receveur** : Le circuit qui reçoit la donnée du bus
- **Maître** : Le circuit qui commence le transfert, génère l'horloge et termine le transfert
- **Esclave** : Le circuit adressé par le maître.

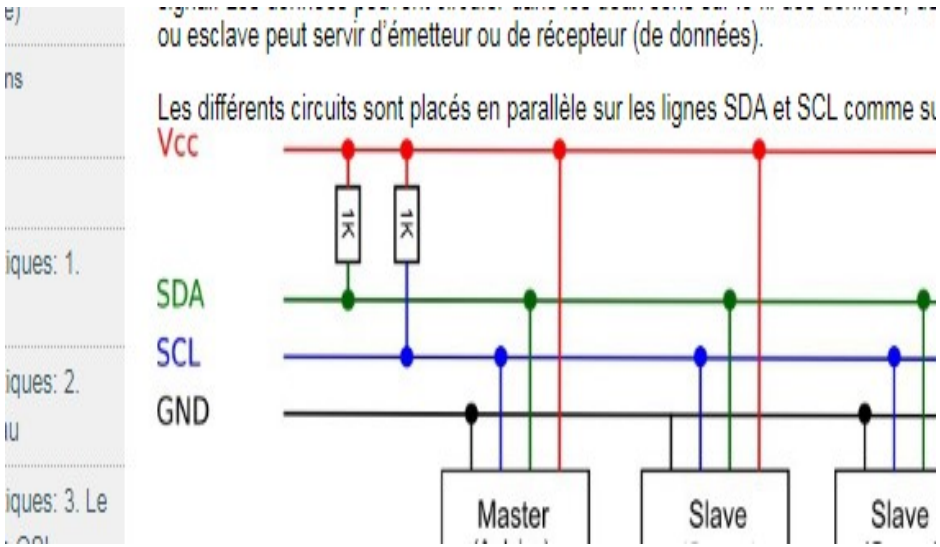


Types de bus : I2C

Caractéristiques

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils :

- ❑ un signal de donnée SDA (Serial Data Line),
- ❑ un signal d'horloge SCL(Serial Clock Line),
- ❑ un signal de référence électrique (Masse).



Ceci permet de réaliser des équipements ayants des fonctionnalités très puissantes (En apportant toute la puissance des systèmes microprogrammés) et conservant un circuit imprimé très simple, par rapport un schéma classique (8bits de données, 16 bits d'adresse + les bits de contrôle

Types de bus : I2C

Principe

Dans le protocole du bus I2C le circuit maître est celui qui demande un transfert d'information sur le bus et qui génère le signal d'horloge qui permet le transfert. Ainsi un circuit adressé est considéré comme un esclave.

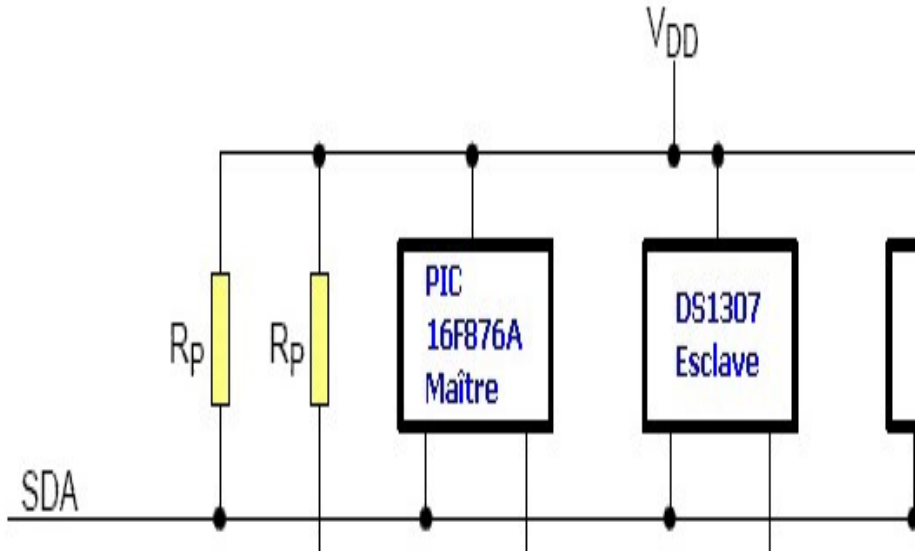


Figure : Principe du bus I2C

Conclusion

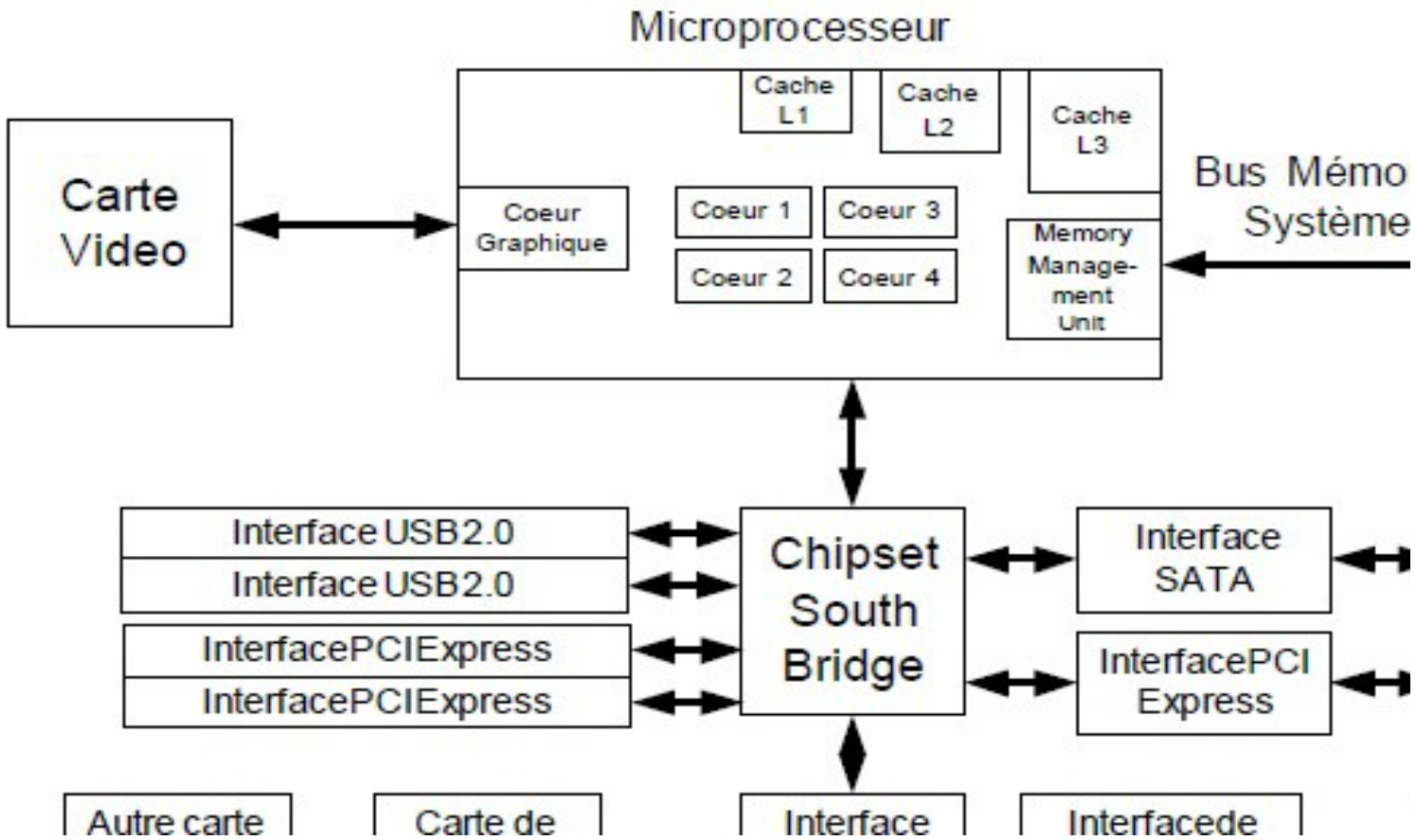


Figure : Architecture de bus d'un Micro-ordinateur 2011

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Plan de Cours

1. Projets réalisés
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. ***Les Entrés/ Sorties***
8. Les interruptions
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

7.1 Introduction
7.2 Ports d'entrées /sorties
7.3 Périphériques d'entrées /sorties
7.4 Transmission série/ parallèle
7.5 Transmission synchrones /asynchrones
7.6 Le protocole
7.7 Normes RS-232 et DB 25

introduction

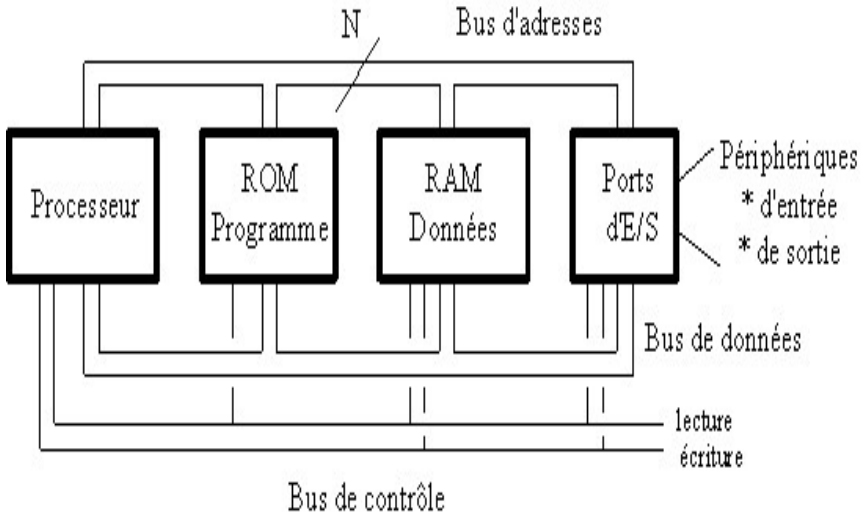
Dans un système à base d'un microprocesseur ou d'un microcontrôleur, on appelle entrées-sorties les échanges d'informations entre le processeur et les périphériques qui lui sont associés. De la sorte, le système peut réagir à des modifications de son environnement. Elles sont parfois désignées par l'acronyme I/O (Input/Output) ou encore E/S pour Entrées/Sorties.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

Ports d'entrées /sorties

Ports d'entrées /sorties

- Les périphériques sont reliés au reste du système par des circuits appelés ports d'entrées et ports de sortie (certains ports peuvent combiner les deux fonctions).



Ports d'entrées /sorties

Ports d'entrées /sorties

- Un port d'entrée est essentiellement composé de tampons trois états. Ceux-ci se comportent comme des interrupteurs électroniques qui font apparaître, au moment voulu, les niveaux logiques du périphérique d'entrée (choisi par le bus d'adresse) sur le bus de données ; ces niveaux seront mémorisés dans un registre du processeur.

- Un port de sortie est essentiellement composé de bascules de type D. Celles-ci se comportent comme des petites mémoires. Leur entrée est reliée au bus de données. Le processeur vient écrire un niveau logique 0 ou 1 dans chacun des bascules. Les sorties des bascules contrôlent les périphériques, généralement via un étage de puissance.

Périphériques d'entrées /sorties

Périphériques d'entrée

Une entrée est un flux de données provenant soit :

- du réseau,
- d'une saisie clavier, d'un mouvement de souris, d'un crayon optique
- ou de tout autre périphérique prévu pour interagir avec un système informatique.

Ces signaux d'entrée génèrent des Interruptions matérielles qui sont traitées en priorité par le gestionnaire d'interruptions du noyau du système d'exploitation.

Périphériques d'entrées /sorties

Périphériques d'entrée

Dans les systèmes informatiques, le choix est bien plus vaste : clavier , souris, crayon optique, numériseur, convertisseurs analogiques/numériques…

Dans les systèmes à microcontrôleurs , tels la machine à laver, on trouve des boutons poussoirs, des commutateurs.

Notons sur le fait que, pour être traités par le processeur, les signaux, quels qu'ils soient, doivent être convertis en signaux logiques compatibles avec le processeur.

Périphériques d'entrées /sorties

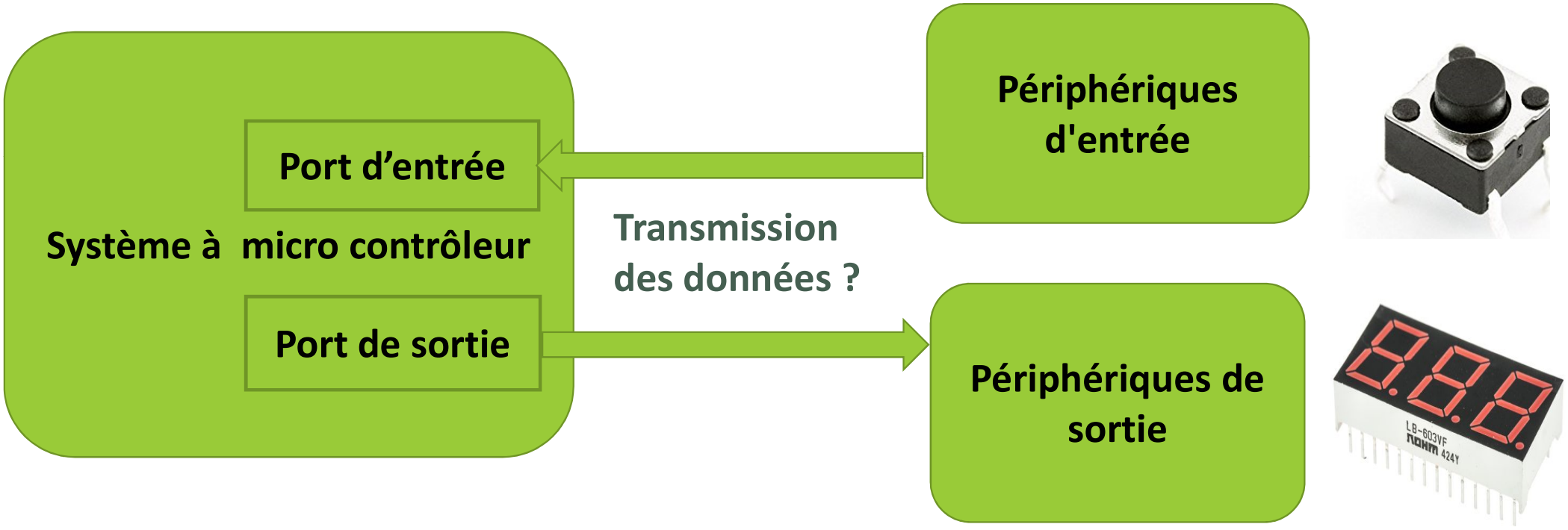
Périphériques de sortie

Une sortie peut être :

- un signal (électrique, onde…) ;
- un flux de données (réseau), une écriture sur disque ou une mise en mémoire ;
- un affichage, un son.

Dans les systèmes informatiques, le choix est vaste : écran pour l'affichage, imprimante pour la production de documents sur papier, convertisseurs numériques/analogiques…

Dans les systèmes à microprocesseurs, on utilise des diodes électroluminescentes LED comme des indicateurs, des afficheurs numériques ou alphanumériques à LED ou à cristaux liquides pour l'affichage des messages du système, des relais (pour commander des charges nécessitant des courants et/ou des tensions élevés)…


Port d'entrée
Système à micro contrôleur
Port de sortie
Transmission des données ?
Périphériques d'entrée
Périphériques de sortie

Transmission série/ parallèle

Transmission série/ parallèle

Un signal numérique transmet généralement plusieurs digits binaires.
Exemple : 01000001 (huit bits).

Dans une transmission numérique on peut envisager deux modes :

- les envoyer tous en même temps sur autant de lignes de transmission.
  C'est le mode parallèle.
- les envoyer l'un après l'autre sur une seule ligne de transmission.
  C'est le mode série.

Transmission série/ parallèle

Transmission Parallèle

- Le bus de communication parallèle utilise un certain nombre de bits transmis simultanément. Un signal d'horloge est nécessaire car l'un des fronts sera utilisé pour indiquer au périphérique quand accepter les données du bus.

- La vitesse de communication est limitée, de même que la longueur du bus.

Transmission série/ parallèle

Transmission Série

- Le bus de communication série transmet d'information via une seule voie. Les bits à transmettre sont stockés dans un registre à décalage qui sera vidé dans le bus à une certaine cadence.

- Si un signal d'horloge est transmit sur un autre fil, le périphérique pourra utiliser un des fonts pour lire le bit présent sur le bus. Cette communication sérielle est dite synchrone.

- En l'absence de signaux d'horloge, la communication sérielle est alors asynchrone.

Transmission Synchrones /Asynchrones

Transmission Synchrones

- Lors de la communication synchrone, le front montant (ou descendant) de l'horloge est utilisé par l'expéditeur du message pour présenter un nouveau bit.
- Le front descendant suivant (ou montant) de l'horloge est utilisé par le récepteur du message pour lire ce bit.
- L'état de l'horloge est maintenu constant entre deux messages.
- La communication synchrone permet de transmettre de très longue trames de données, puisque la synchronisation entre l'émetteur et le récepteur est assurée par le signal d'horloge.

Transmission Synchrones /Asynchrones

Transmission Asynchrones :

- Lors de la communication asynchrone, le message commence en envoyant un certain nombre de bits de démarrage. L'état de ces bits diffère de l'état de repos entre deux transmissions.

- Le récepteur utilise ces bits pour sa synchronisation. L'horloge du récepteur fonctionne de façon indépendante de celle de l'émetteur. Il est important qu'ils soient ajustés à la même cadence (on nomme cette cadence le baud rate).

- Les fréquences des deux horloges n'étant virtuellement jamais exactement identiques, le nombre de bits d'une trame est très limité

Transmission Synchrones /Asynchrones

Erreurs de transmission

- Lorsque l'on transmet un message, il peut être bon de vérifier que le message sera reçu correctement ou minimalement être en mesure de savoir si le message est reçu correctement, sans erreurs.

- Dans la communication sérielle asynchrone on a adopté une approche nommée « bit de parité ». Le message contient un certain nombre de bits à un niveau logique 0 les autres étant à 1. On compte le nombre de bits à 1 dans le message et on ajuste le bit de parité à 1 ou à 0 selon que la parité soit paire ou impaire.

Transmission Synchrones /Asynchrones

Bit de parité - Exemple

- Message 0110 0001 (comportant un nombre impair de 1)

- Si parité paire, il faut un nombre pair de 1, donc le bit de parité sera fixé à 1. (message 0110 0001, le nb 1 = 3, donc il faut fixé le bit de parité a 1 pour que la somme des 1 soit paire).

- Si parité impaire, il suffit de fixer le bit de parité à 0, car le nombre de bits à 1 est déjà impair. (message 0110 0001, le nb 1 = 3, donc il faut fixé le bit de parité a 0 pour que la somme des 1 soit impaire).

Transmission Synchrones /Asynchrones

Bit de parité - Exemple

- Message transmit (avec parité paire) :
  - 10010010 1
  - Nombre de bits à 1 : 4.

- Message reçu :
  - 10011010 1
  - Nombre de bits à 1 : 5 (impair ! ! ! Erreur)

- C'est donc une façon très simple de détecter une erreur.

Le Protocole

Protocole

On nomme **protocole** les conventions qui facilitent une communication sans faire directement partie du sujet de la communication elle-même.

Un protocole informatique est un ensemble de règles qui régissent les échanges de données ou le comportement collectif de processus ou d'ordinateurs en réseaux ou d'objets connectés.

Un protocole de communication est un ensemble de contraintes permettant d'établir une communication entre deux entités : Internet Protocol, Suite des protocoles Internet, Protocole réseau par exemple ;

Protocole de communication PC-PC

Le Protocole

Protocole

Pour qu'un message soit correctement reçu et interprété il faut respecter quelques règles :

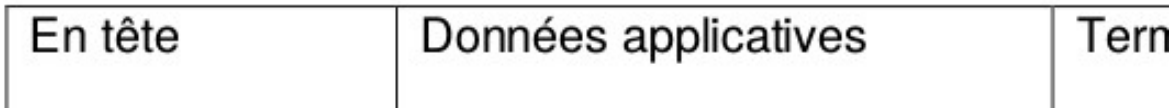
- s'adresser au bon destinataire,
- lui indiquer que la communication débute,
- parler la même « langue » etc...

Protocole de communication PC-PC

Le Protocole

La trame

Une information ne sera donc jamais envoyée seule mais sera toujours précédée et terminée par des données (binaires) dites « de service ». Celles qui précédent l'information à transmettre (donnée(s) applicative(s)) constituent l'en-tête, celles qui la suivent correspondent au terminateur.



Mdèle d'une trame

Norme RS232

RS 232

La liaison série à la norme RS 232 permet la réalisation d'une liaison simple à mettre en oeuvre entre deux équipements. Elle est de type ***asynchrone***, c'est à dire qu'elle ne transmet pas de signal d'horloge, les deux équipements doivent être configurés avec une vitesse de transmission identique. Ils doivent par ailleurs utiliser le même protocole.

Schéma fonctionnel RS-232

Les niveaux logiques niveau 0 = +12 V
niveau 1 = -12 V

Norme RS232

Protocole de transmission

1. Longueur des mots : 7 bits (ex : caractère ascii) ou 8 bits
2. La vitesse de transmission : les différentes vitesses de transmission sont réglables à partir de 1200 bauds (bits par seconde) de la façon suivante : 1200 bds, 2400 bds, 4800 bds, 9600 bds.

Débit (bit/s)	Longueur (m)
2 400	60
4 800	30
9 600	15
19 200	7,6
38 400	3,7
56 000	2,6

Norme RS232

Protocole de transmission

3. Parité : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux types de parité.

4. Bit de start : la ligne au repos est à l'état logique 1(-12V), pour indiquer qu'un mot va être transmis, la ligne passe à l'état bas (+12V) avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.

5. Bit de stop : après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes d'horloge selon le nombre de bits de stop.

Norme RS232

Protocole de transmission

Pour établir une communication via RS-232, il est nécessaire de définir le protocole utilisé : notamment, le débit de la transmission, le codage utilisé, le découpage en trame, etc. le flux en trames d'un caractère ainsi constituées :

- ✓ 1 bit de départ ;
- ✓ 7 à 8 bits de données ;
- ✓ 1 bit de parité optionnel ;
- ✓ 1 ou 2 bits d'arrêt.

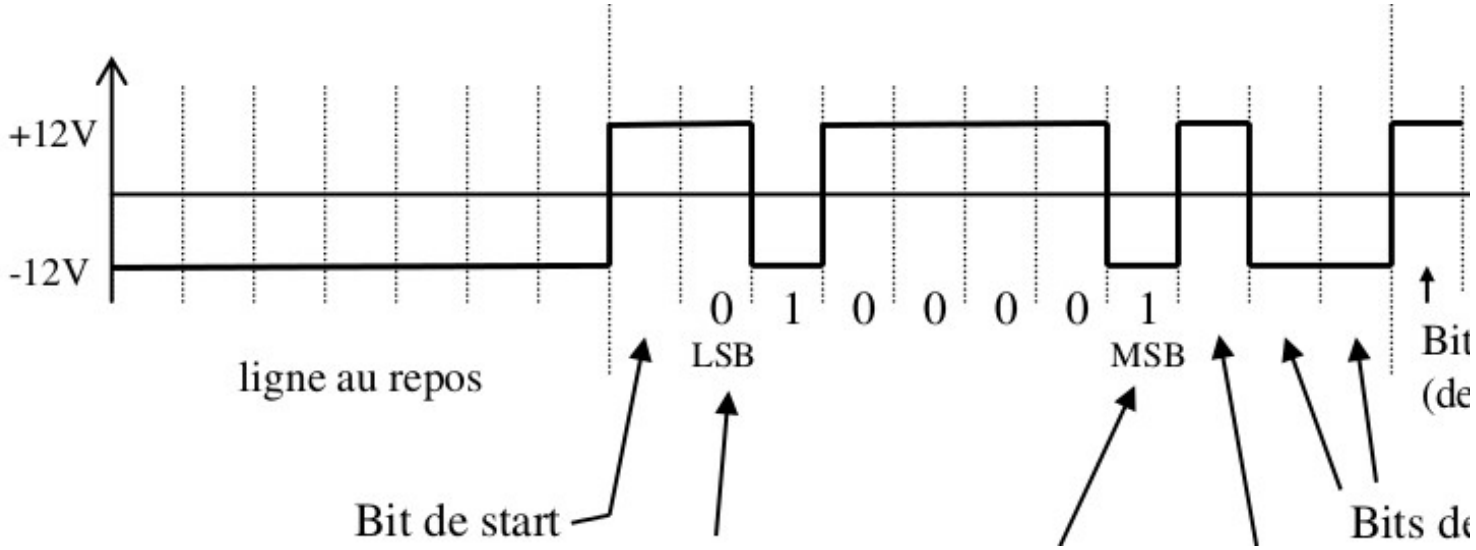
Le bit de départ a un niveau logique "0" tandis que le bit d'arrêt est de niveau logique "1". Le bit de donnée de poids faible est envoyé en premier suivi des autres.

Norme RS232

Protocole de transmission

Le bit de start apparait en premier dans la trame puis les données (poids faible en premier), la parité éventuelle et le (les) bit(s) de stop.

Exemple : Soit à transmettre en parité paire, avec 2 bits de stop, le caractère B dont le codage ASCII est B ’1000010’ la trame sera la suivante :



Norme RS232

Protocole de transmission

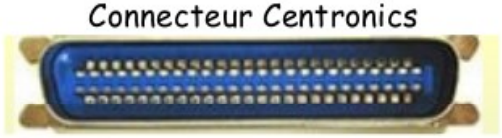
Signal		Origin		DE-9
Name	Abréviation	DTE	DCE	
Transmitted Data	TxD	•		3
Received Data	RxD		•	2
Data Terminal Ready	DTR	•		4
Data Carrier Detect	DCD		•	1
Data Set Ready	DSR		•	6
Ring Indicator	RI		•	9
Request To Send	RTS	•		7
Clear To Send	CTS		•	8
Signal Ground	G	common		5

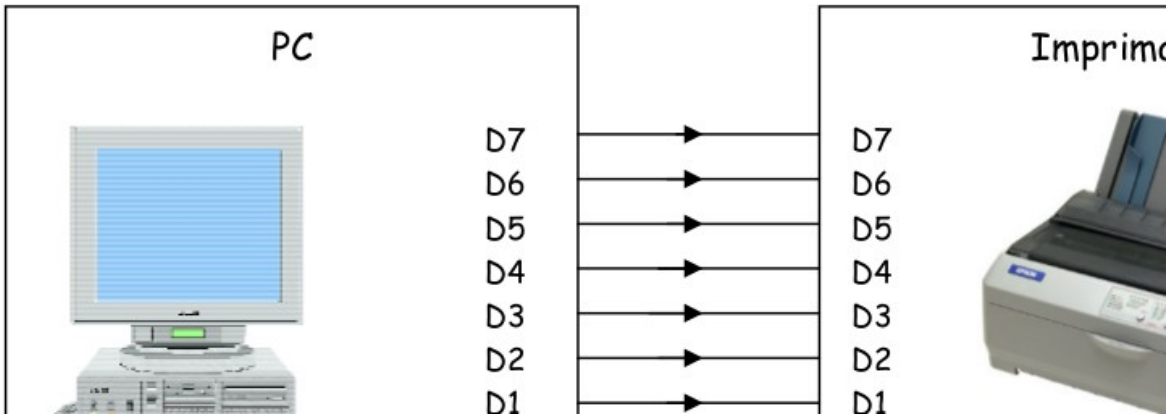
Un connecteur DE-9,

DB25

DB25

Ce mode utilise généralement un connecteur SUB D 25 côté PC et « Centronics »côté imprimante :



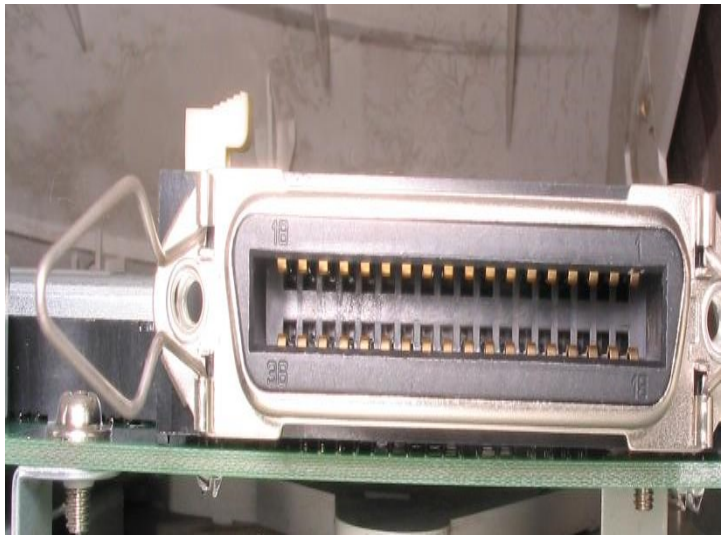


Liaison DB25 PC/Imprimente

DB25

DB25

Centronics mâle



Centronics femelle

DB 25 parallèle femelle

DB25

DB25

- Les bits sont envoyés simultanément sur N voies différentes (une voie étant par exemple un fil , un câble ou tout autre support physique.
- Il faut autant de lignes de transmission que de bits à transmettre : D 0 à D 7, plus une équipotentielle zéro (la référence de tension) GND, plus un signal dit d'échantillonnage STR/, plus un signal d'acquittement ACK/.
- La transmission est très rapide puisque les 8 bits de données sont transmis en même temps.

Remarque : Etant donné que les fils conducteurs sont proches sur une nappe, il existe des perturbations (notamment à haut débit) dégradant la qualité du signal... Cette liaison limite donc la longueur du câble de transmission, généralement de 2 à 3 mètres.

DB25

DB25

Numéro	Nom	Désignation
1	_STR - Strobe	Balayage
2	D0 - Data bit 0	Bit de données 0
3	D1 - Data bit 1	Bit de données 1
4	D2 - Data bit 2	Bit de données 2
5	D3 - Data bit 3	Bit de données 3
6	D4 - Data bit 4	Bit de données 4
7	D5 - Data bit 5	Bit de données 5
8	D6 - Data bit 6	Bit de données 6
9	D7 - Data bit 7	Bit de données 7 (poids fort)

Numéro	Nom	Désignation
10	ACK - Acknowledgement	Acquittement
11	Busy	Occupé (lecture des données)
12	Paper Out	Plus de papier
13	Select	Sélection
14	Auto feed	Saut de page
15	Error	Erreur
16	Reset	Réinitialisation
17	Select Input	Sélection de l'entrée
18-25	GND	Masse

Plan de Cours

1. Projets réalisés
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. ***Les interruptions***
9. Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560

8.1 Présentation

- ***Déroulement d'une interruption***
- ***Déroulement d'une interruption***
- ***Interrupt Service Routine***
- ***Interruptions imbriquées et priorités***
- ***Contrôleur d'interruptions***

8.2 Les types d'interruption

- ***Interruption système***
- ***Exceptions***
- ***Interruptions matérielles***
- ***Interruptions logicielles***

Présentation

Problématique

Un système informatique n'est utile que s'il communique avec l'extérieur. L'objectif est de pouvoir prendre connaissance que le périphérique sollicité le processeur. Cette sollicitation arrive de façon totalement asynchrone.

Solutions

Deux modes sont possibles :

- Scrutation : (polling) Permet d'intérroger régulièrement les périphériques afin de s'avoir si une nouvelle donnée est présente.
- Interruption : Permet au périphérique lui-même de faire signe au processeur de sa présence.

Présentation

Scrutation

- Coûteux en temps (multiplier par le nombre de périphérique à interroger).
- Implémentation : Appel classique à une fonction dans le programme.

Interruption

- Demande à l'initiative du périphérique
- Prise en compte rapide de l'évènement
- Implémentation : Interruption asynchrone d'un programme puis retour au même endroit à la fin du traitement

Présentation

Définition

Une interruption est un arrêt temporaire de l'exécution normale d'un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption).

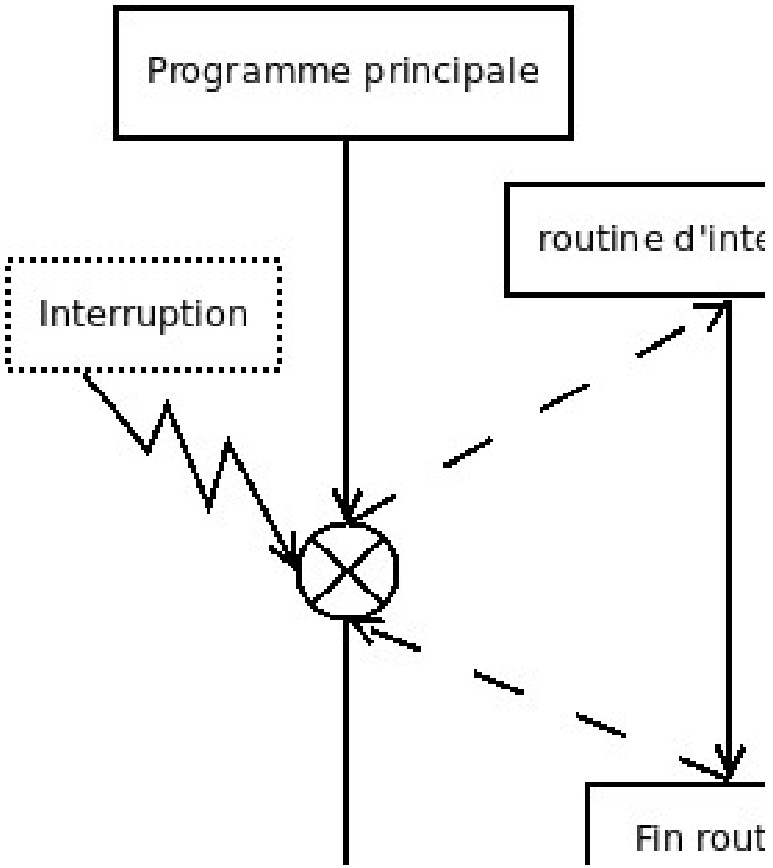


Schéma d'une interruption

Présentation

Dans son acception la plus stricte, le terme ne désigne que des interruptions dont l'exécution est provoquée par des causes externes au programme :
avancement d'une horloge, signalisation de la complétion d'un transfert de données, positionnement des têtes de lecture/écriture, etc.

Cependant, on l'utilise aussi pour désigner des exceptions, c'est-à-dire des arrêts provoqués par une condition exceptionnelle dans le programme ***(instruction erronée, accès à une zone mémoire inexistante, calcul arithmétique incorrect, appel volontaire au système d'exploitation, etc.***).

On parle alors parfois d'interruptions asynchrones pour désigner celles résultant d'un événement externe, et d'interruptions synchrones pour désigner les exceptions provoquées par le déroulement du programme.

Présentation

- Une interruption interrompt l'exécution des instructions par le micro processeur.

- Lors d'une interruption :

  - L'exécution du programme principal est suspendue ;
  - Une sous-routine traitant l'interruption est exécutée ;
  - Puis le programme principal est continué.

Remarque

La différence entre interruptions et un branchement ou un appel de fonction ?

- les interruptions peuvent survenir n'importe quand pendant l'exécution

Déroulement d'une interruption

Lorsqu'une interruption survient il se produit :

1. Terminer l'instruction en cours.

2. Vérifier si l'interruption peut être traitée.

3. L'interruption peut être masquée (désactivée) ou ne pas être exécutée parce qu'une autre interruption de priorité supérieure ou égale est en cours. L'interruption est mise de côté pour être exécutée ultérieurement (lorsqu'elle sera réactivée ou lorsque l'interruption de priorité plus grande finira).

Déroulement d'une interruption

4. Sauvegarder (sur la pile) :
   - l'adresse de retour
   - les registres (en pratique, pas tous)
   - les drapeaux de l'ALU

5. Déterminer l'adresse de la routine qui traitera l'interruption (Interrupt Service Routine, ou "ISR"). Cette routine est en mémoire code.

6. Exécuter cette routine

7. PC = Adresse de l'ISR, donc l'exécution d'instructions continue dans l'ISR.

Interrupt Service Routine

Les routines de traitement d'interruptions ("Interrupt Service Routines — ISR) sont en mémoire, elles permettent de répondre aux questions suivantes :

- Quelle routine exécuter pour quelle interruption ?
- à quelle adresse est cette routine ?

Table des vecteurs d'interruption(VI)

Table des vecteurs d'interruption	
# d'INT	Contenu
0	Adresse de la routine à exécuter quand l'INT
1	Adresse de la routine à exécuter quand l'INT
2	Adresse de la routine à exécuter quand l'INT
3	Adresse de la routine à exécuter quand l'INT

Table des vecteurs d'interruption

Interruptions imbriquées et priorités

- Qu'arrive-t-il si une interruption survient lorsqu'on traite une interruption ?
- Cela dépend de la priorité
  - Si la priorité de la nouvelle interruption est plus élevée :
    - ❑ On interrompt l'exécution et on traite cette nouvelle interruption
  - Si la priorité de la nouvelle interruption est moins élevée :
    - ❑ On attend que le traitement de l'interruption à plus haute priorité soit terminé, et on traite cette nouvelle interruption par la suite

Interruptions imbriquées et priorités

Remarque 1

Sauvegarder les adresses de retour et certains registres sur la pile permet d'imbriquer les interruptions comme on imbrique des fonctions.

Remarque 2

Une instruction est nécessaire pour indiquer la fin de l'interruption. Cette instruction, "retour d'interruption", permet de :

- Récupère les valeurs des registres sauvegardés sur la pile,
- Récupère l'adresse de retour et les drapeaux sauvegardés sur la pile,
- Met PC = adresse de retour de l'interruption

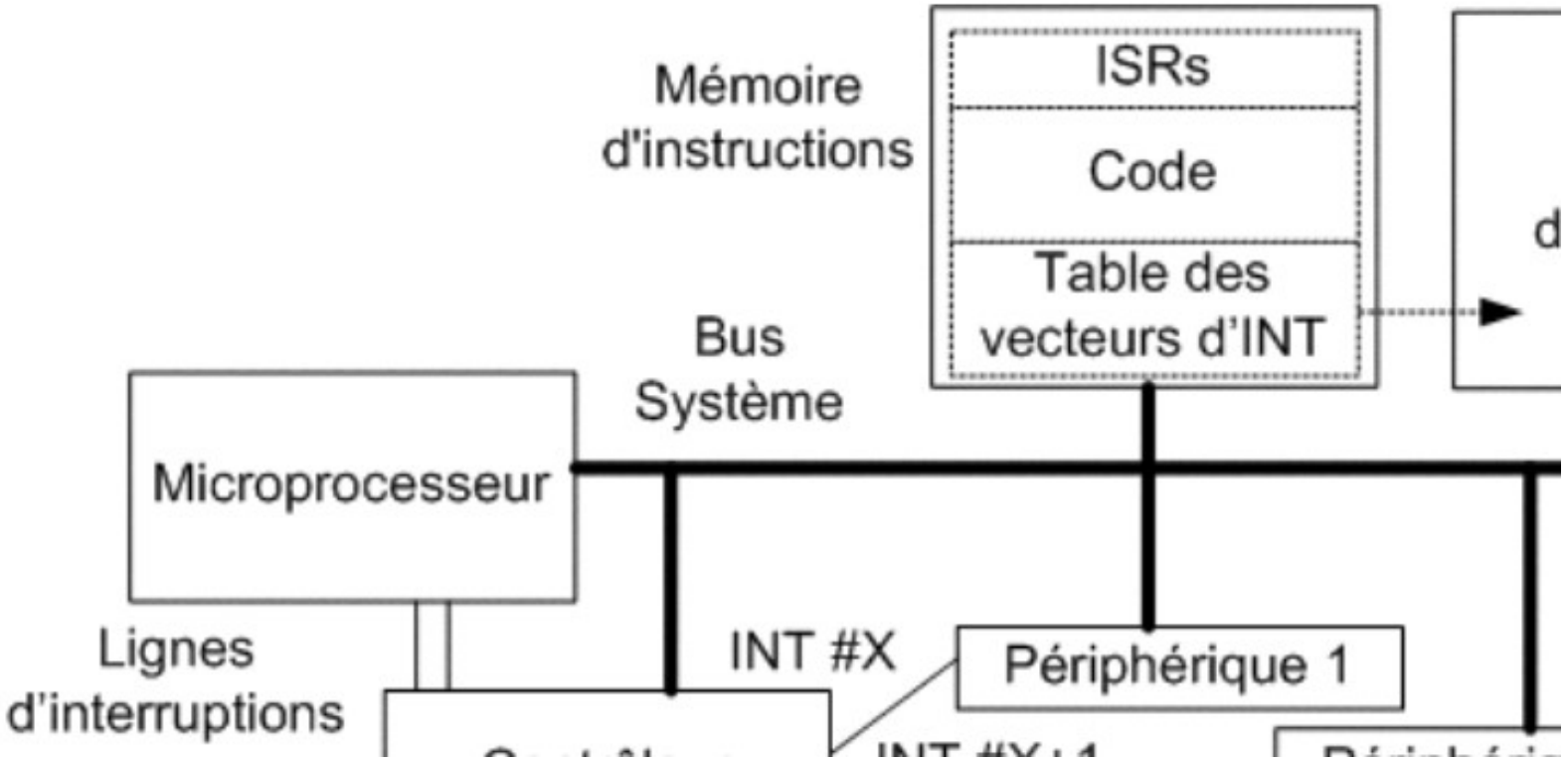
Contrôleur d'interruptions

Les interruptions sont gérées par le contrôleur d'interruption.

Le contrôleur d'interruptions :

- reçoit les signaux d'interruptions
- peut activer (masquer) ou désactiver certaines interruptions.
- modifier la priorité des interruptions.
- signale les interruptions au microprocesseur à l'aide de fils dédiés à cette fin.
- peut être configuré via des instructions dans la mémoire

Contrôleur d'interruptions



Controleur d'interruption

Les types d'interruptions

- Système : reset, NMI (non-maskable Interrupt), faute matérielle générale, etc.

- Exception : le processeur peut générer des interruptions s'il n'est pas capable de lire ou d'exécuter une instruction (opcode invalide, division par 0, mémoire protégée, etc).

- Matérielles : générées par les périphériques.

- Logicielles : il y a une instruction qui permet de générer une interruption dans tous les jeux d'instructions.

- Il y a aussi des interruptions pour le mode "debug"...

Les types d'interruptions

Interruption système

- L'interruption reset est l'interruption système la plus commune. Cette interruption peut survenir pour plusieurs raisons :
  - mise sous tension, activation de la broche reset du microprocesseur, instruction reset, etc.

- Lors d'un reset, toutes les autres interruptions sont ignorées.

- L'interruption NMI (Non Maskable Interrupt) est aussi fréquente dans les systèmes ordinateurs. NMI est une interruption que ne peut pas être désactivée par le logiciel et qui est souvent utilisée pour détecter une faute d'alimentation ou une mise hors tension.

Les types d'interruptions

Interruption d'Exceptions

- Les exceptions surviennent quand un évènement logiciel spécial arrive. Cet évènement logiciel empêche le microprocesseur d'exécuter l'instruction en cours pour diverses raisons :
  - instruction invalide, division par 0, référence à une adresse invalide, accès invalide à une adresse protégée, faute matérielle, etc.

- Les exceptions ont un très haut niveau de priorité parceque le microprocesseur est dans une impasse : il ne peut exécuter l'instruction en cours en raison d'une erreur de programmation !

Les types d'interruptions

Interruption matérielles

- Les interruptions matérielles sont générées par les périphériques.

- La plupart des périphériques ont une ligne de contrôle reliée au contrôleur d'interruptions qui leur permet de signaler un événement.

- Lors d'une interruption de périphérique, le microprocesseur obtient automatiquement le # de l'interruption du NVIC et utilise ce numéro pour trouver l'ISR à exécuter à partir de la table des vecteurs d'interruptions.

Les types d'interruptions

Interruption logicielles

• Les interruptions logicielles sont des interruptions "provoquées" par le programmeur. Le programmeur utilise une instruction qui déclenche une interruption.

• Les interruptions logicielles ont un effet similaire à un appel de fonction avec une différence fondamentale :

- l'adresse de la fonction appelée est dans la table des vecteurs d'interruption plutôt qu'être une adresse relative au programme.

• Les interruptions logicielles servent souvent à appeler des fonctions du système d'exploitation dont l'adresse est inconnue du programmeur, mais gérée par le système d'exploitation (grâce à la table des vecteurs d'interruption).

Plan de Cours

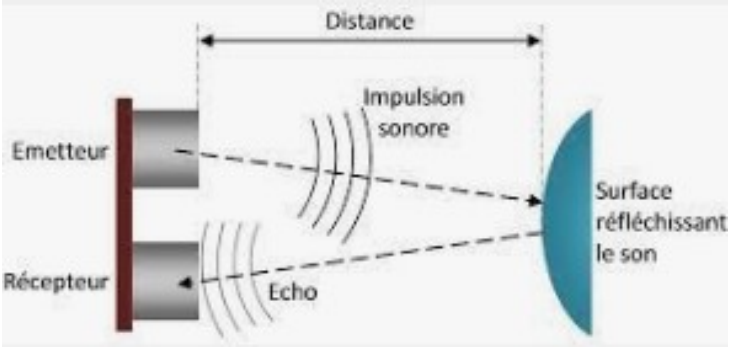
1. Projets réalisés
2. Introduction
3. Types d'architecture
4. Les Processeurs
5. Type des mémoires
6. Les Bus de communication
7. Les Entrés/ Sorties
8. Les interruptions
9. **Microcontrôleur ATmega328 / ATmega32u4 / ATmega2560**

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

Exercice 1 :

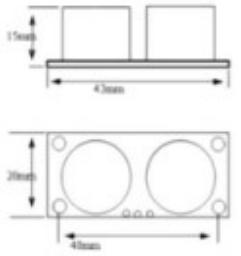
Émetteur - récepteur ultrason, pour mesurer des distances (Ultrasonic sensor HC_SR04). Un capteur à ultrasons émet à intervalles réguliers de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l'air à la vitesse du son (0.340 [mm / micro-secondes]). Lorsqu'elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d'écho au capteur.

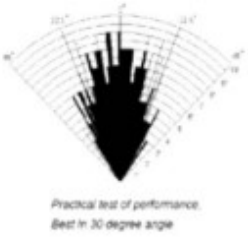


VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND

Features:

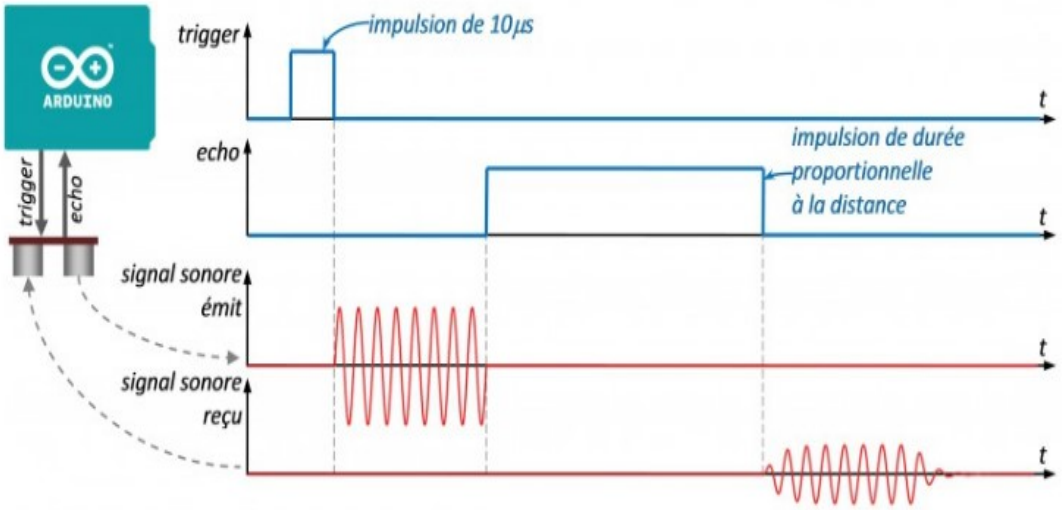
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm





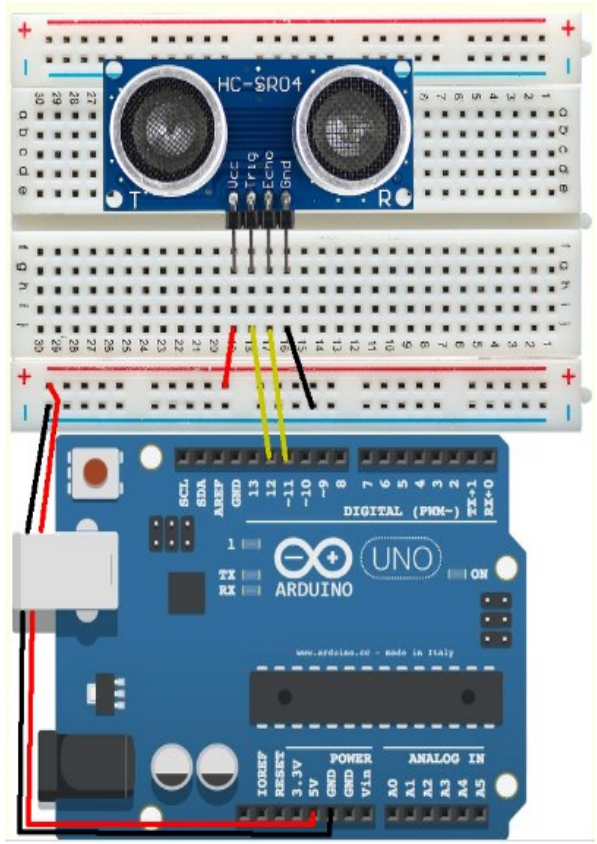
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

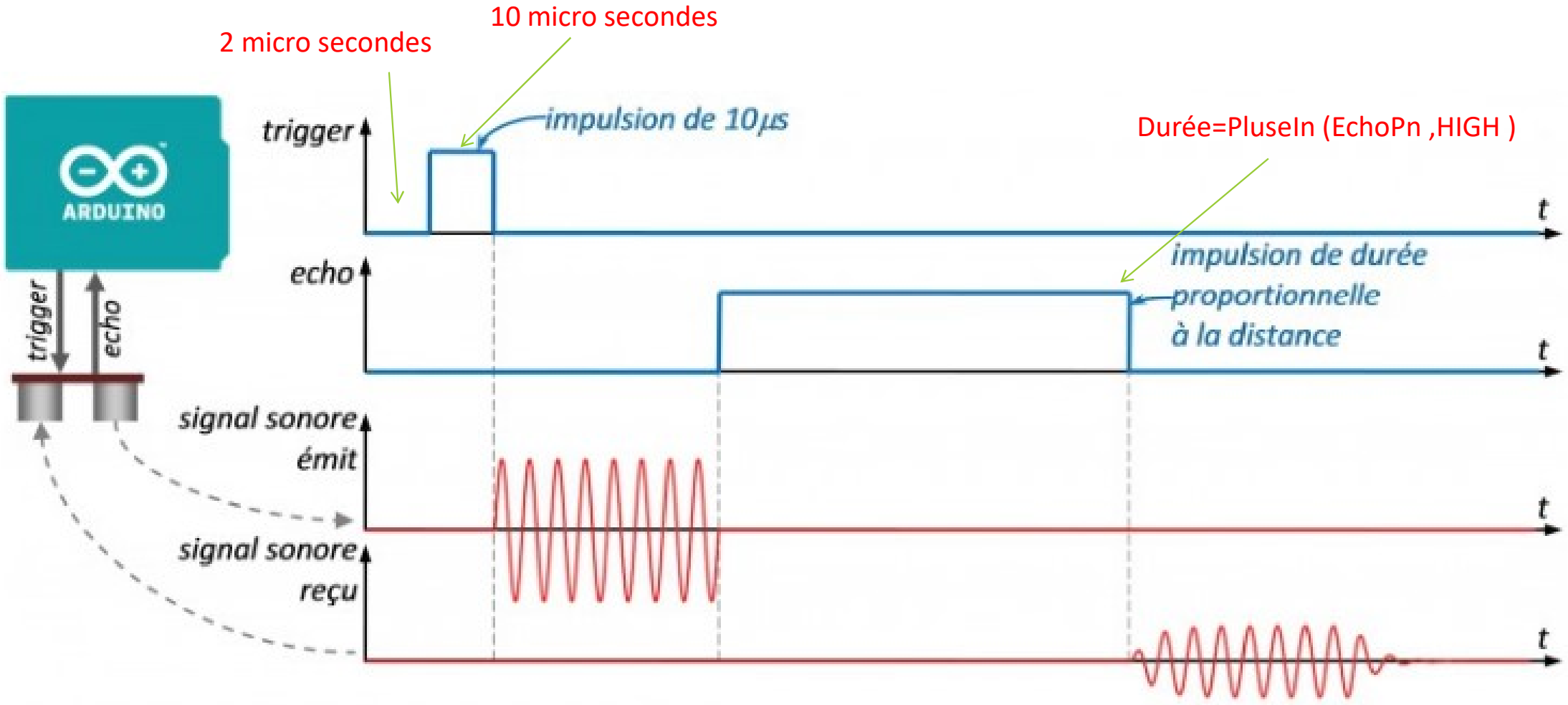


Remarque : si le signal retour n'est pas détecté, la durée du signal « echo » est limitée à 36 ms environ (cela dépend des modèles), soit 6m environ.

Circuit réalise



TD


2 micro secondes
10 micro secondes
ARDUINO
trigger
echo
trigger
impulsion de 10µs
t
Durée=PluseIn (EchoPn ,HIGH)
echo
impulsion de durée
proportionnelle
à la distance
t
signal sonore
émit
t
signal sonore
reçu
t

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 1 : Solution (Affiche dans le serial)

```
// defines pins numbers
const int trigPin = 12;
const int echoPin = 11;

// defines variables
long duration; // durée en micro-secondes
int distance;  // distance en centimètres

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}
```

```
void loop() {
// Envoie le signal de début de mesure sur le "trigPin"
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Met le "trigPin" à l'état haut durant 10 micro-secondes.
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Lecture sur "l'echoPin" du temps d'aller-retour, en microseconds
duration = pulseIn(echoPin, HIGH);

// Calcule de la distance entre le sonnar et l'objet qui a réfléchi l'impulsion sonor
distance= duration*0.340/2;  // 0.340 [mm / micro-secondes] est la vitesse du son dans l'air

// Affiche le résultat
Serial.print("temps [micro-secondes] = ");
Serial.print(duration);
Serial.print("  Distance [mm] : ");
Serial.println(distance);

delay(1000); // Attente
} // loop
```

TD
Exercice 1 : Solution (Version pour l'afficheur LCD)

```
#include <LiquidCrystal.h> // includes the LiquidCrystal Library
LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
const int trigPin = 12;
const int echoPin = 11;
long duration;
int distanceCm, distanceInch;
void setup() {
lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies
                 // the dimensions (width and height) of the display
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
```

```
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
distanceInch = duration*0.0133/2;
lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD wil
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distanceCm); // Prints the distance value from the sensor
lcd.print(" cm");
delay(10);
lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print(" inch");
delay(10);
}
```

TD
Exercice 2 :

Lecture de luminosité en utilisant une photo-résistance.

- Lecture d'une tension variable, réglée par la luminosité arrivant sur une photo résistance.
- Allume des LEDs parmis 3, en fonction de la luminosité.
- Donne sur les LEDs un codage binaire de l'intensité lue.

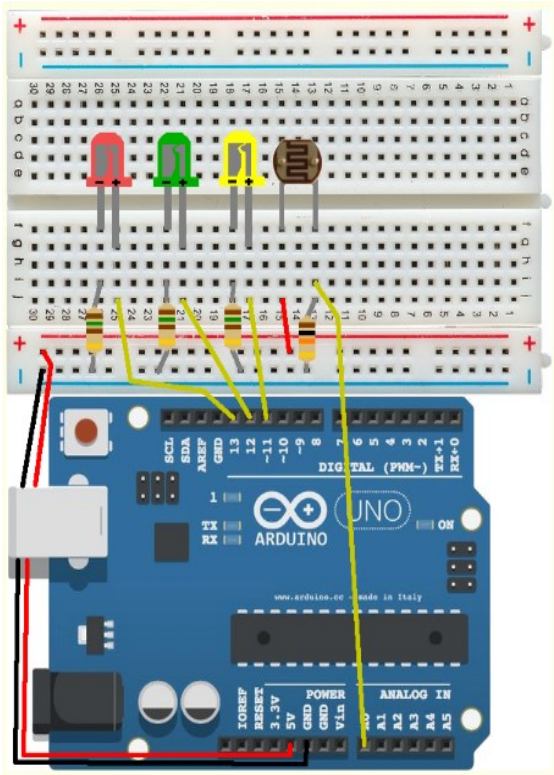
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

Exercice 2 :

Une photorésistance (également appelée cellule photoconductrice ou cellule photoéléctrique) est un composant électronique dont la résistivité varie en fonction de la quantité de lumière incidente : plus elle est éclairée, plus sa résistivité baisse.

Circuit réalise



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 3 :

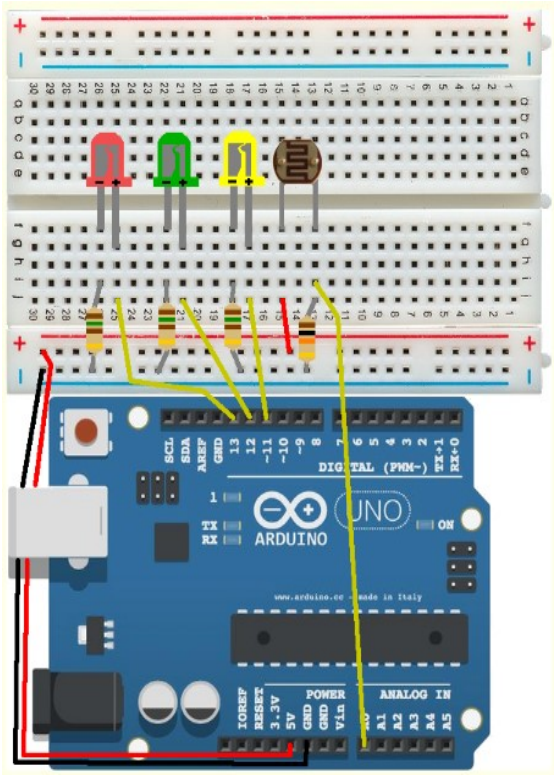
Dans la pin A0 :

La tension reçu entre 0 et 5V → une valeur numérique entre 0 et 1024

```
// Convertie la valeur lue, qui est entre 0 et 1023 en un nombre entre 0 et 7.
nConvertValue = map(nSensorValue, 0, 1023, 0, 7);
```

N°	Pin 13	Pin 12	Pin 11
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Circuit réalise



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 2 : Solution

```
int nSensorValue = 0;    // Valeur lue sur le port analogique A0
int nConvertValue = 0;   // Valeur lue, convertit dans un autre intervalle

void setup() {

// Initialise les PINs digital 13, 12 et 11 comme OUTPUT
// La PIN 13 a une LED connectée sur la plupart des cartes Arduino
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
pinMode(11, OUTPUT);
// Pour TESTER
//Serial.begin(115200);  // Initialise la communication série à une haute vitesse.
                       // La vitesse de 9600 bauds est souvent utilisée par défaut.
} // setup
```

```
void loop() {
//===========
nSensorValue = analogRead(A0); // Lecture de la tension sur le port analogique A0
                               // La tension varie entre 0 et 5 Volts, correspondant
                               // aux nombres entre 0 et 1023.

// Convertie la valeur lue, qui est entre 0 et 1023 en un nombre entre 0 et 7.
nConvertValue = map(nSensorValue, 0, 1023, 0, 7);

// Test le niveau global d'intensité
if (nConvertValue >= 4) { digitalWrite(13, HIGH); }
else { digitalWrite(13, LOW); }

if ((nConvertValue % 4) >= 2) { digitalWrite(12, HIGH);}
else {digitalWrite(12, LOW);}

if ((nConvertValue % 2) >= 1) { digitalWrite(11, HIGH); }
else { digitalWrite(11, LOW); }

// affiche la valeur convertie  POUR TESTER
// Serial.println(nConvertValue);  delay(500);
} // loop
```

TD
Exercice 3 :

Mesure du temps de passage d'un objet devant une photorésistance Le temps sera transmis à travers le port série.

Mesure durant combien de microsecondes la photorésistance recoit une luminosité faible.

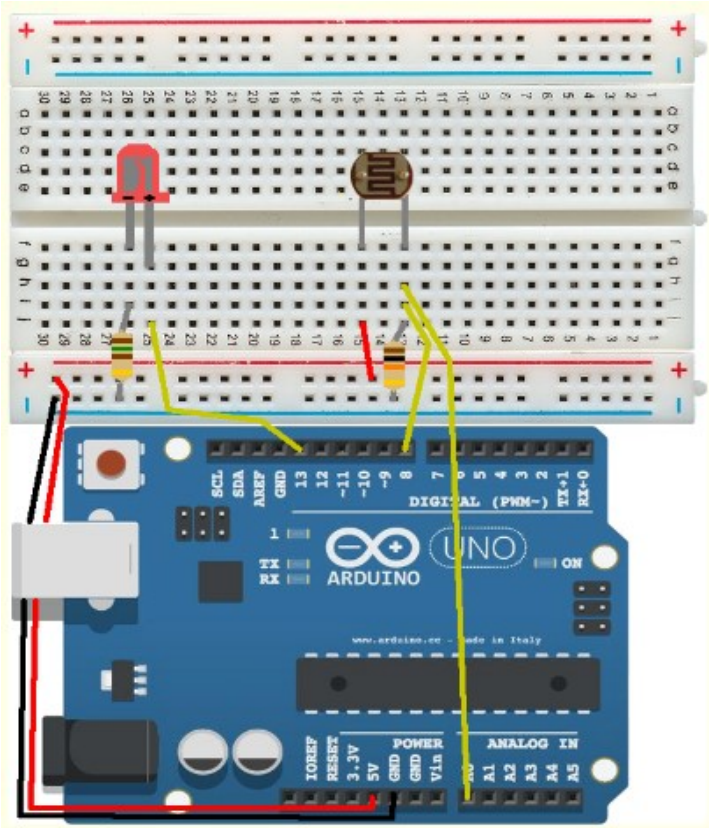
L'utilisation de la lecture analogique est lente, elle prend environ 100 micro-secondes, donc la précision est limitée.

Seul l'entrée analogique A0 est utilisée.

La PIN 13 est utilisée pour indiquer que la mesure est en cours.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 3 :



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 3 : solution

```
int nSensorValue = 0;    // Valeur lue sur le port analogique A0
int nSensorValueOld = 0; // Dernière valeur lue sur le port analogique A0
unsigned long lwTimeStart = 0; // Temps en micros secondes du départ de bouton appuyé
unsigned long lwTimeStop = 0;  // Temps en micros secondes d'arret du bouton

void setup() {
// Initialise la PIN digital 13 comme OUTPUT
pinMode(13, OUTPUT);
digitalWrite(13, LOW);
nSensorValueOld = analogRead(A0); // Lecture de la tension sur le port analogique A0
                                  // La tension varie entre 0 et 5 volts, correspondant
                                  // aux nombres     entre 0 et 1023.
Serial.begin(115200);  // Initialise la communication série à une haute vitesse.
                       // La vitesse de 9600 bauds est souvent utilisée par défaut.
} // setup
```

```
void loop() {
nSensorValue = analogRead(A0); // Lecture de la tension sur le port analogique A0
                               // La tension varie entre 0 et 5 volts, correspondant
                               // aux nombres entre 0 et 1023.

if ((nSensorValue < 500) & (nSensorValueOld > 500)) {
   // Début de la mesure de temps
   lwTimeStart = micros();
   nSensorValueOld = nSensorValue;  // < 500, donc une mesure est en cours
   digitalWrite(13, HIGH); // Allume la LED pour indiquer qu'une mesure est en cours
   }

if ((nSensorValue > 550) & (nSensorValueOld < 500)) {
   // Fin de la mesure de temps
   lwTimeStop = micros();
   nSensorValueOld = nSensorValue; // > 500, donc une prochaine mesure peut etre faite
   digitalWrite(13, LOW); // Eteint la LED, ce qui indique la fin de mesure.

   // Affiche le temps
   Serial.println(lwTimeStop - lwTimeStart);
   }
} // loop
```

TD
Exercice 4 :

Mesure le temps en micro secondes durant lequel un bouton est pressé.

Lecture de l'état du bouton connecté à la PIN 12

S'il est pressé, allume la LED connectée à la PIN 13

Mesure en micro secondes durant combien de temps il est pressé affiche dans le "moniteur série" (Ctrl+Maj+M) ce temps.

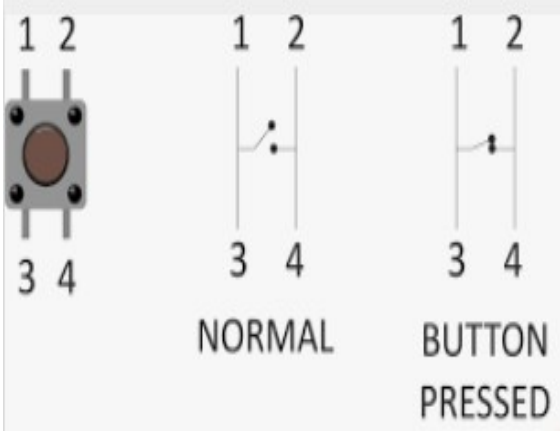
Avec une petite amélioration en ajoutant une attente lorsque le bouton a été pressé, pour éviter les rebonds et compter à double une pression sur le bouton.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

Exercice 4 :

Fonctionnement d'une bouton poussoir



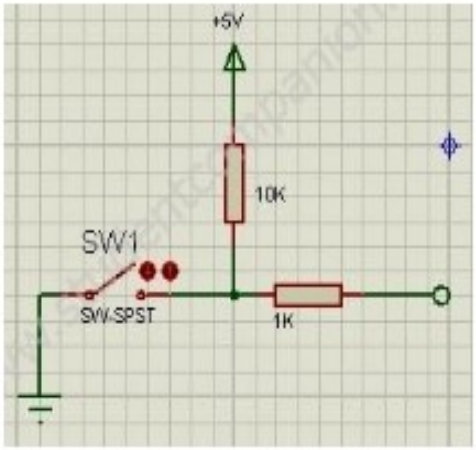


Figure 1: A switch with a Pull-up resistor

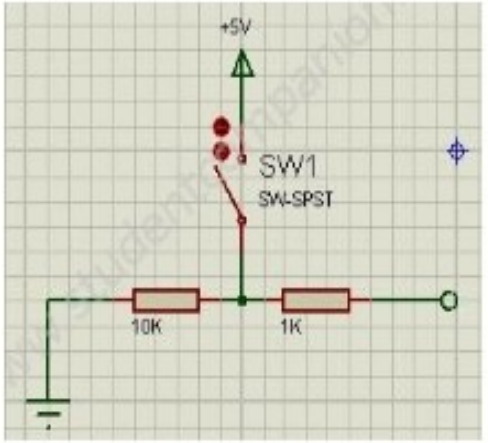


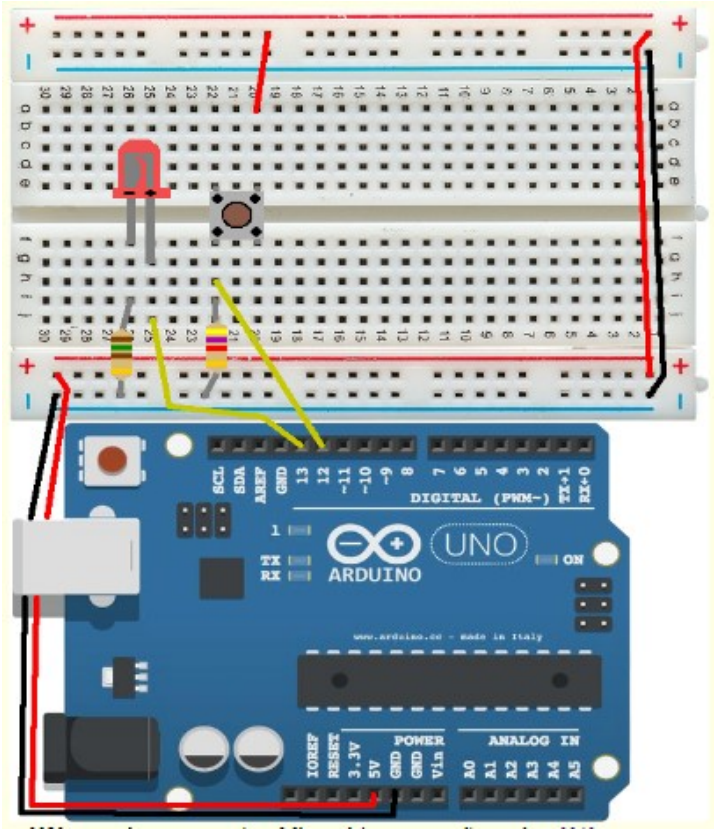
Figure 2: A switch with a Pull-down resistor

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

Exercice 4 :

Circuit réalise



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD

Exercice 4 : solution

```
int nEtatLast = 0; // mémorise le dernier état du bouton
unsigned long lwTimeStart = 0; // Temps en micros secondes du départ de bouton appuyé
unsigned long lwTimeStop = 0;  // Temps en micros secondes d'arret du bouton

void setup() {
// Initialise la PIN digital 13 comme OUTPUT
// La PIN 13 a une LED connectée sur la plupart des cartes Arduino
// Initialise la PIN digital 12 comme INPUT
pinMode(13, OUTPUT);
pinMode(12, INPUT);
digitalWrite(13, LOW); // set the LED off
Serial.begin(115200);  // Initialise la communication série à une haute vitesse.
                       // La vitesse de 9600 bauds est souvent utilisée par défaut.
} // setup
```

```
void loop() {

if (digitalRead(12) == 1) {
  // On a pressé sur le bouton qui met la PIN 12 à l'état haut (HIGH)
  // Test qu'il n'était pas déjà pressé avant.
  if (nEtatLast == 0) {
    lwTimeStart = micros();
    nEtatLast = 1; // Mémorise qu'on a pressé sur le bouton
    delay(1); // attente d'une milliseconde, pour éviter les rebonds

    digitalWrite(13, HIGH);  // set LED on
    }
  }
else { // bouton relaché
  if (nEtatLast == 1) {
    // On vient de relacher le bouton
    lwTimeStop = micros();
    nEtatLast = 0; // Mémorise qu'on a relaché le bouton

    Serial.println(lwTimeStop - lwTimeStart); // Affiche en micro secondes le temps durant lequel le bouton était pressé

    digitalWrite(13, LOW);    // set the LED off
    }
  }
} // loop
```

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 5 :

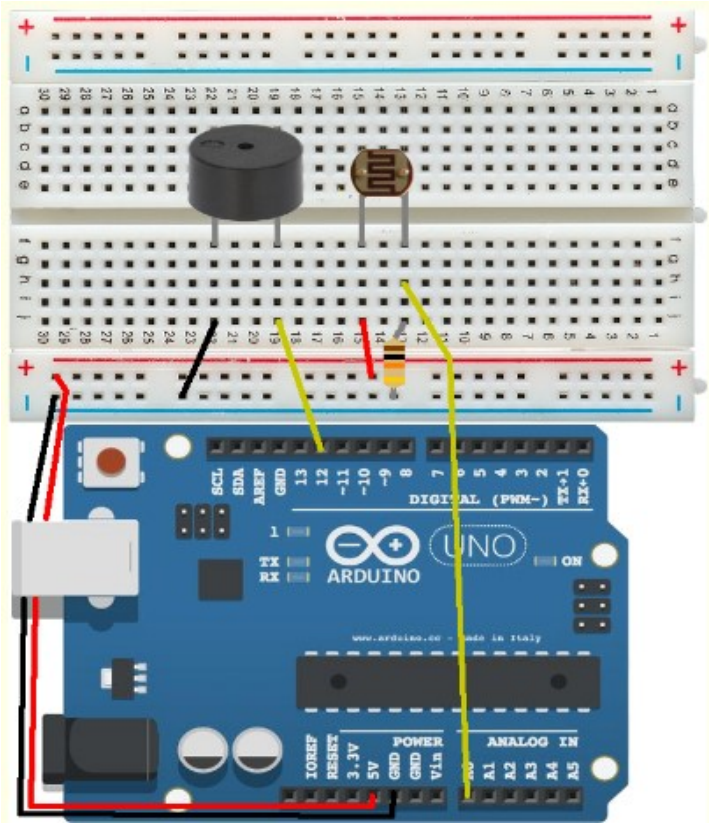
Un buzzer émet un son à une fréquence dépendante de la valeur lue sur l'entrée A0, déterminée par la lumière arrivant sur une photo-résistance.

Un buzzer (anglicisme) ou bipeur est un élément électromécanique ou piézoélectrique qui produit un son caractéristique quand on lui applique une tension : le bip. Certains nécessitent une tension continue, d'autres nécessitent une tension alternative.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 5 :

Circuit réalise



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 5 :

```
int nSensorValue = 0;   // Valeur lue sur le port analogique A0
unsigned long lwTimeLast = 0; // Dernier temps lu en micros secondes
unsigned long lwTimeNow = 0;  // Temps lu en micros secondes
void setup() {
pinMode(12, OUTPUT);
lwTimeLast = micros(); // Lecture du temps au départ
lwTimeNow = micros(); // Lecture du temps au départ
} // setup

void loop() {
nSensorValue = analogRead(A0); // Lecture de la tension sur le port analogique A0
                               // La tension varie entre 0 et 5 volts, correspondant
                               // aux nombres entre 0 et 1023.
// Attente, jusqu'à ce que le temps atteigne une demi période
while (lwTimeNow - lwTimeLast < 5*nSensorValue) lwTimeNow = micros();
digitalWrite(12, HIGH);
lwTimeLast = lwTimeNow; // mémorise le changement d'état

// Nouvelle lecture de l'entrée analogique, pour régler la période.
nSensorValue = analogRead(A0);

// Attente, jusqu'à ce que le temps atteigne une demi période
while (lwTimeNow - lwTimeLast < 5*nSensorValue) lwTimeNow = micros();
digitalWrite(12, LOW);
lwTimeLast = lwTimeNow; // mémorise le changement d'état
} // loop
```

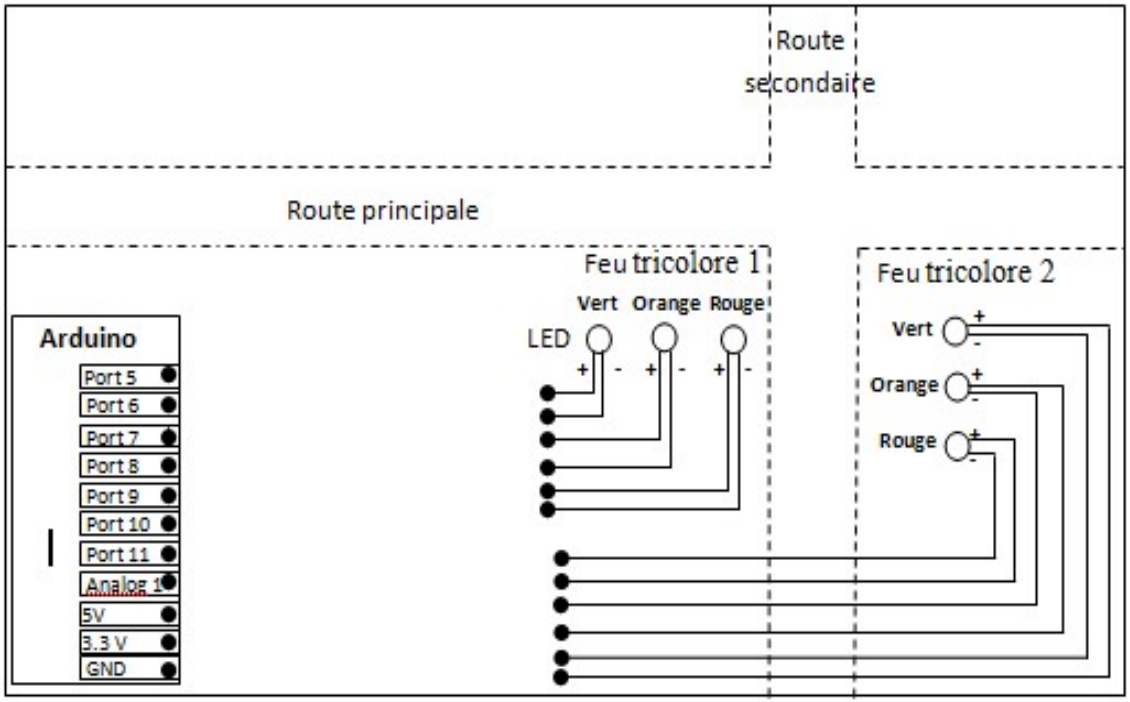
Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 6 :

Nous désirons réaliser deux feux tricolores synchronisés qui permettent la régulation de la circulation d'un carrefour à deux voies. Les durées d'allumage des deux feux tricolores sont décrites comme suit :

La durée d'allumage du feu vert est égale à 20 secondes ;
La durée d'allumage du feu orange vert est égale à 5 secondes ;
La durée d'allumage du feu rouge est égale à 25 secondes.

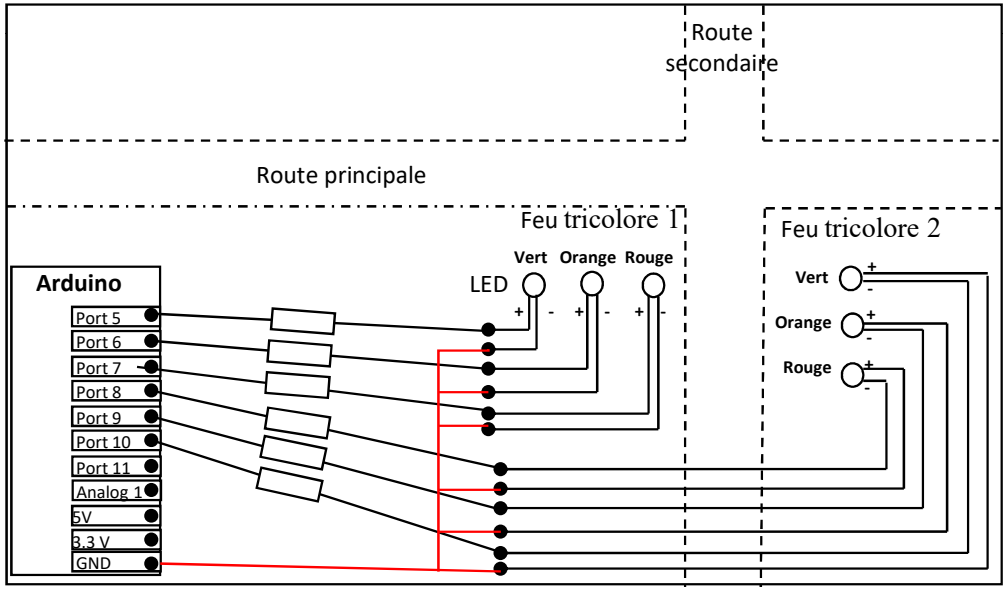
1. Compléter le circuit électronique des feux tricolores
2. Ecrire un programme Arduino qui permet de répondre à ce cahier de charges.



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 6 : solution

1) Compléter le circuit électronique des feux tricolores :



TD

Exercice 6 : solution

2) Ecrire un programme Arduino qui permet de répondre à ce cahier de charges.

```
#define LED_Rouge_Tricolore_1  5
#define LED_Orange_Tricolore_1  6
#define LED_Vert_Tricolore_1  7
#define LED_Rouge_Tricolore_2  8
#define LED_Orange_Tricolore_2  9
#define LED_Vert_Tricolore_2  10
void setup() {
pinMode(LED_Rouge_Tricolore_1,OUTPUT);
pinMode(LED_Orange_Tricolore_1,OUTPUT);
pinMode(LED_Vert_Tricolore_1,OUTPUT);
pinMode(LED_Rouge_Tricolore_2,OUTPUT);
pinMode(LED_Orange_Tricolore_2,OUTPUT);
pinMode(LED_Vert_Tricolore_2,OUTPUT);
}

delay(5000);
}
```

```
void loop() {
digitalWrite(LED_Rouge_Tricolore_1,LOW);
digitalWrite(LED_Orange_Tricolore_1,LOW);
digitalWrite(LED_Vert_Tricolore_1,HIGH);
digitalWrite(LED_Rouge_Tricolore_2,HIGH);
digitalWrite(LED_Orange_Tricolore_2,LOW);
digitalWrite(LED_Vert_Tricolore_2,LOW);
delay(20000);
digitalWrite(LED_Rouge_Tricolore_1,LOW);
digitalWrite(LED_Orange_Tricolore_1,HIGH);
digitalWrite(LED_Vert_Tricolore_1,LOW);
digitalWrite(LED_Rouge_Tricolore_2,HIGH);
digitalWrite(LED_Orange_Tricolore_2,LOW);
digitalWrite(LED_Vert_Tricolore_2,LOW);
delay(5000);
digitalWrite(LED_Rouge_Tricolore_1,HIGH);
digitalWrite(LED_Orange_Tricolore_1,LOW);
digitalWrite(LED_Vert_Tricolore_1,LOW);
digitalWrite(LED_Rouge_Tricolore_2,LOW);
digitalWrite(LED_Orange_Tricolore_2,LOW);
digitalWrite(LED_Vert_Tricolore_2,HIGH);
delay(20000);
digitalWrite(LED_Rouge_Tricolore_1,HIGH);
digitalWrite(LED_Orange_Tricolore_1,LOW);
digitalWrite(LED_Vert_Tricolore_1,LOW);
digitalWrite(LED_Rouge_Tricolore_2,LOW);
digitalWrite(LED_Orange_Tricolore_2,HIGH);
digitalWrite(LED_Vert_Tricolore_2,LOW);
delay(5000);
}
```

TD
Exercice 7 :

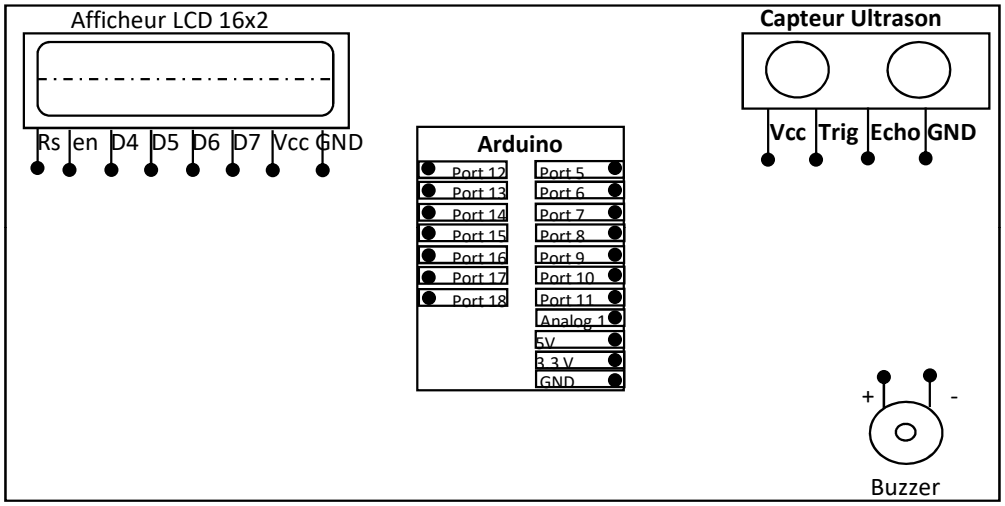
Nous désirons réaliser un appareil à base d'Arduino permet de mesurer la distance parcourue par un objet en mouvement sur une seconde afin de calculer sa vitesse en cm/s. L'appareil est doté d'un afficheur LCD 16x2, d'un capteur Ultrason, d'un Arduino et d'un Buzzer.

Arduino envoie au capteur Ultrason par un port numérique une courte impulsion (10µs environ) à l'entrée « trigger » du capteur, cela déclenche l'émission d'un signal sonore très court (8 oscillations environ). Lorsque ce signal est parti, la sortie « echo » du capteur passe à l'état HIGH. Dès que le signal sonore réfliche par l'objet cible, il est détecté par le capteur dont la sortie « echo » repasse à LOW. Le temps allez routour de la propagation de l'implusion est obtenu par la fonction Temps_propagation = pulseIn(Numero de la PN Echo, HIGH), la vitesse du son égale à 0.340 [mm / micro-secondes]. La valeur de la ditance entre l'objet et le capteur ultrason doit etre afficher au premier ligne de l'afficheur LCD comme suite : Ditance : 56 cm. La vitesse de l'objet sera affichée à la deuxième ligne de l'afficheur LCD comme suite : Vitesse : 5.2 cm/s. Un bip sonneur doit être déclenche une fois la distance entre la cible et l'objet sera inferieur à 10 cm.
Compléter le circuit électronique des feux tricolores :

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 7 :

1. Compléter le circuit électronique des feux tricolores :



2. A partir de circuit proposé, remplir le tableau suivant :

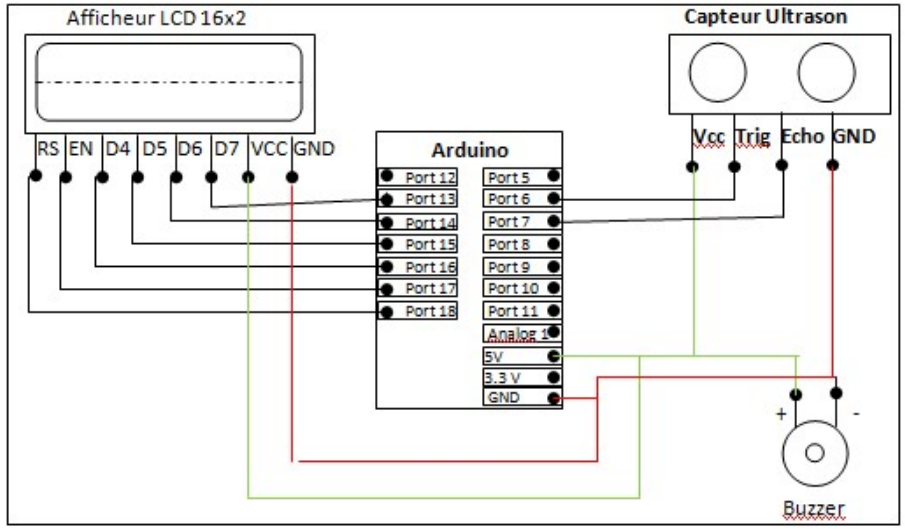
	Ports														
	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Analog 1
Input															
Output															

3. Ecrire un programme Arduino qui permet de répondre à ce cahier de charges.

Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 7 :

1. Compléter le circuit électronique des feux tricolores :



Mr. MEGNAFI Hicham (ESSA-Tlemcen)

TD
Exercice 7 :

2. A partir de circuit proposé, remplir le tableau suivant :

	Ports														
	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Analog 1
Input			x												
Output		X							x	x	x	x	x	x	

TD
Exercice 7 :

3. Ecrire un programme Arduino qui permet de répondre à ce cahier de charges.

```
#include<LiquidCrysal.h>
LiquidCrysal lcd ();//rs, enable, d4,d5,d6,d7
int trigPin,echoPin;
float duration,vitesse,D_entre_implustion,delta_D;
void setup() {
lcd.begin(16,2);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, OUTPUT);
}
void loop() {
digitalWrite(trigPin,LOW);
delayMicroseconds(2);
digitalWrite(trigPin,HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,LOW);

duration1 = pulseIn(echoPin,HIGH);
distance1 = duration1*0.340/2;

digitalWrite(trigPin,LOW);
delayMicroseconds(D_entre_implustion);
digitalWrite(trigPin,HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,LOW);
```

```
duration2 = pulseIn(echoPin,HIGH);
distance2 = duration2*0.340/2;
delta_D = distance2 - distance1;
vitesse = delta_D/10;

lcd.setCursor(0,0);
lcd.print("Ditance : ");
lcd.print(distance);
lcd.print("cm");
lcd.setCursor(0,1);
lcd.print("Vitesse :");
lcd.print(vitesse);
lcd.print("cm/s");
}
```

Fin