

# Entrées & Sorties

[vberry@lirmm.fr](mailto:vberry@lirmm.fr)


# Plan



© Can Stock Photo

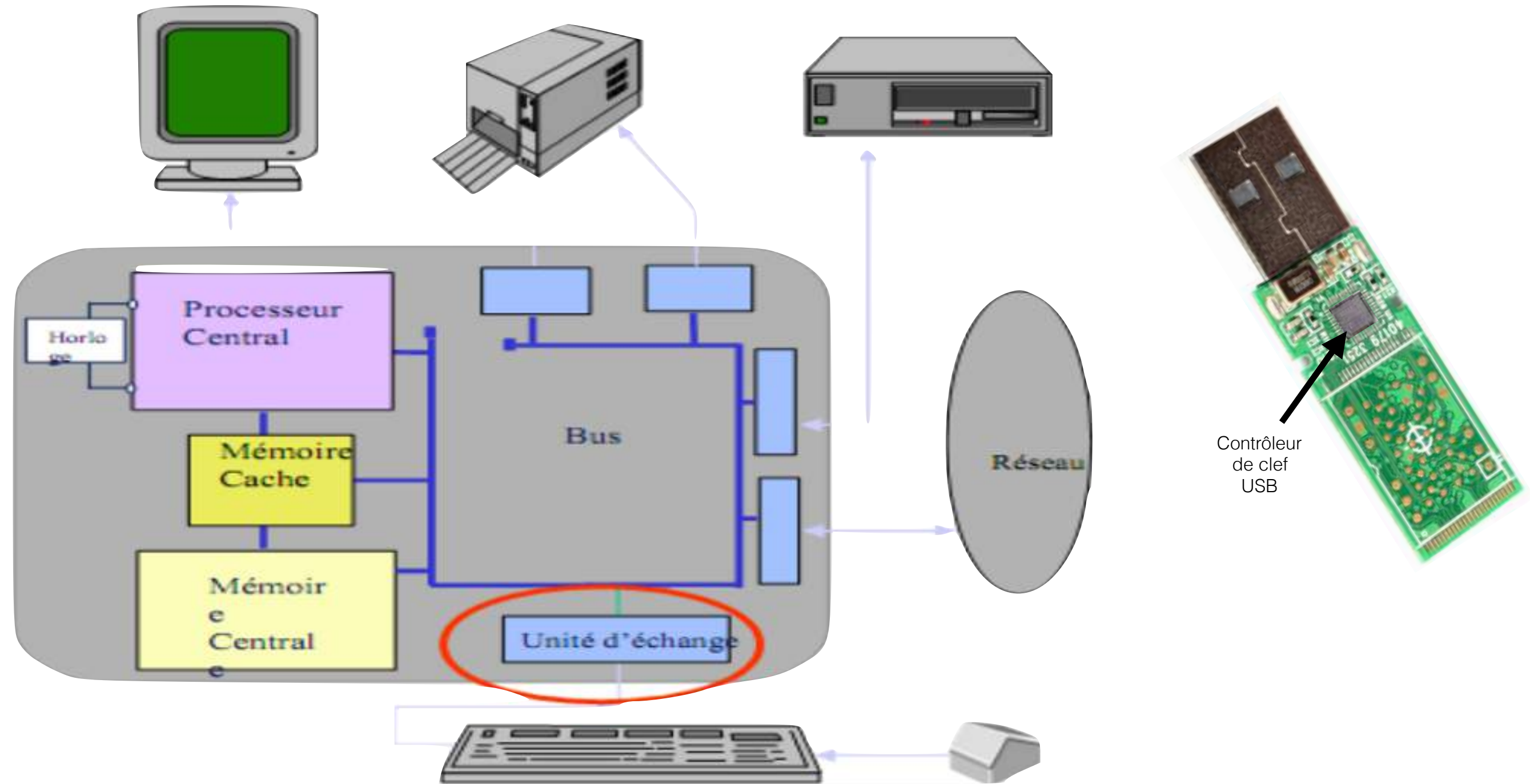
- **Les entrées / sorties :**
  - Quel enjeu ?
  - Quel composants logiciels et matériels sont en jeu ?
  - Par quels protocoles s'effectuent les E/S ?
  - Comment gérer les requêtes concurrentes ?
- **Les bus**
  - Quelles caractéristiques ?
  - Quel protocole de communication ?
  - Exemple du bus I2C (raspberry)

# PÉRIPHÉRIQUES ENTRÉES/SORTIES

- ❖ Entrées/Sorties (E/S) désigne l'ensemble des transferts de données qui permettent au processeur de communiquer avec les périphériques.
  - **Entrées** : données envoyées par un périphérique à destination de l'unité centrale.
  - **Sorties** : données émises par l'unité centrale à destination d'un périphérique.
- ❖ Un **périphérique** est un matériel électronique pouvant être raccordé à un ordinateur par l'intermédiaire de l'une de ses interfaces d'entrée-sortie (interface série, parallèle, USB, etc.).

Listez 5 périphériques d'un ordinateur

# Unité d'échange (UE) ou Contrôleur d'E/S

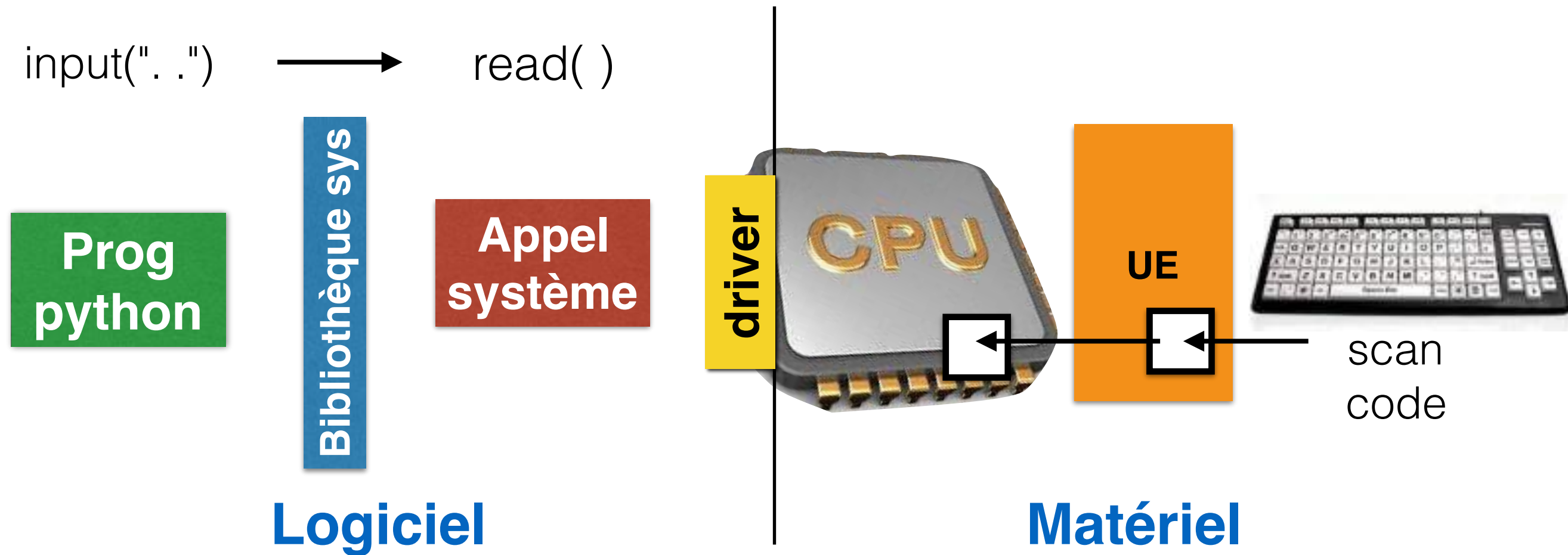


Un périphérique d'E/S contient en fait deux parties : (1) un **appareil** ou dispositif (clavier, écran, disque...) et (2) un **contrôleur de périphérique** aussi appelé l'**Unité d'Echange (UE)**

# CONTRÔLEUR D'ENTRÉE/SORTIE

- ❖ Le contrôleur sert d'interface entre le périphérique et le processeur :
  - Il reçoit les requêtes du processeur et les transforme en commandes pour le périphérique,
    - Les principales requêtes qu'un processeur envoie à un périphérique sont « lecture » et « écriture ».
    - Selon les périphériques, ces requêtes peuvent correspondre à des commandes simples ou complexes.
- ❖ Et réciproquement, il envoie les requêtes du périphérique au processeur.

# Chaîne de transmission de l'information



De nombreux intermédiaires sont impliquées dans la réalisation d'une entrée ou d'une sortie



# Pilote des E/S

- Le **pilote** (ou **driver**) d'E/S est un **programme** du SE responsable de la gestion d'un type particulier de périphérique.
- Il traite les interruptions émises par celui-ci, traite les cas d'erreur. Il est capable d'ordonnancer les accès au périphérique
- C'est lui qui traduit les appels systèmes (`read`, ...) en commandes spécialisées pour le type de périphérique qu'il contrôle
- En Unix tout périphérique est traité comme un fichier. Les claviers sont gérés en mode « caractère » :

```
vberry@mouse:/dev$ ls -l input/
total 0
crw-rw---- 1 root input 13, 64 Aug 29 09:13 event0
crw-rw---- 1 root input 13, 65 Aug 29 09:13 event1
crw-rw---- 1 root input 13, 63 Aug 29 09:13 mice
```



# Protocoles d'E/S

- Le SE. dispose de plusieurs protocoles pour communiquer avec un périphérique :
- Les trois principales techniques sont :

**(A) E/S programmées**

**(B) E/S par interruptions**

**(C) E/S par accès direct à la mémoire (DMA : Direct Memory Access).**



périphérique

- L'essentiel est d'épargner autant que possible le CPU : il fonctionne beaucoup plus rapidement que les périphériques (x1000)



CPU

# a) Protocole avec attente active

- Cas où l'UE du périphérique ne sait pas délivrer d'informations (avertir) sur son état
- Pour savoir si l'unité d'échange est prête à recevoir ou délivrer une nouvelle donnée, le processeur doit régulièrement lire le contenu du registre d'état (RE) de l'unité d'échange.

## **prog du pilote :**

lire le RE

tant que (nb\_car > 0) faire

    tant que périphérique occupé faire  
        lire le RE

    écrire/lire un caractère

    dans le registre de données

    nb\_car = nb\_car - 1

➡ Le processeur est occupé durant toute l'entrée sortie

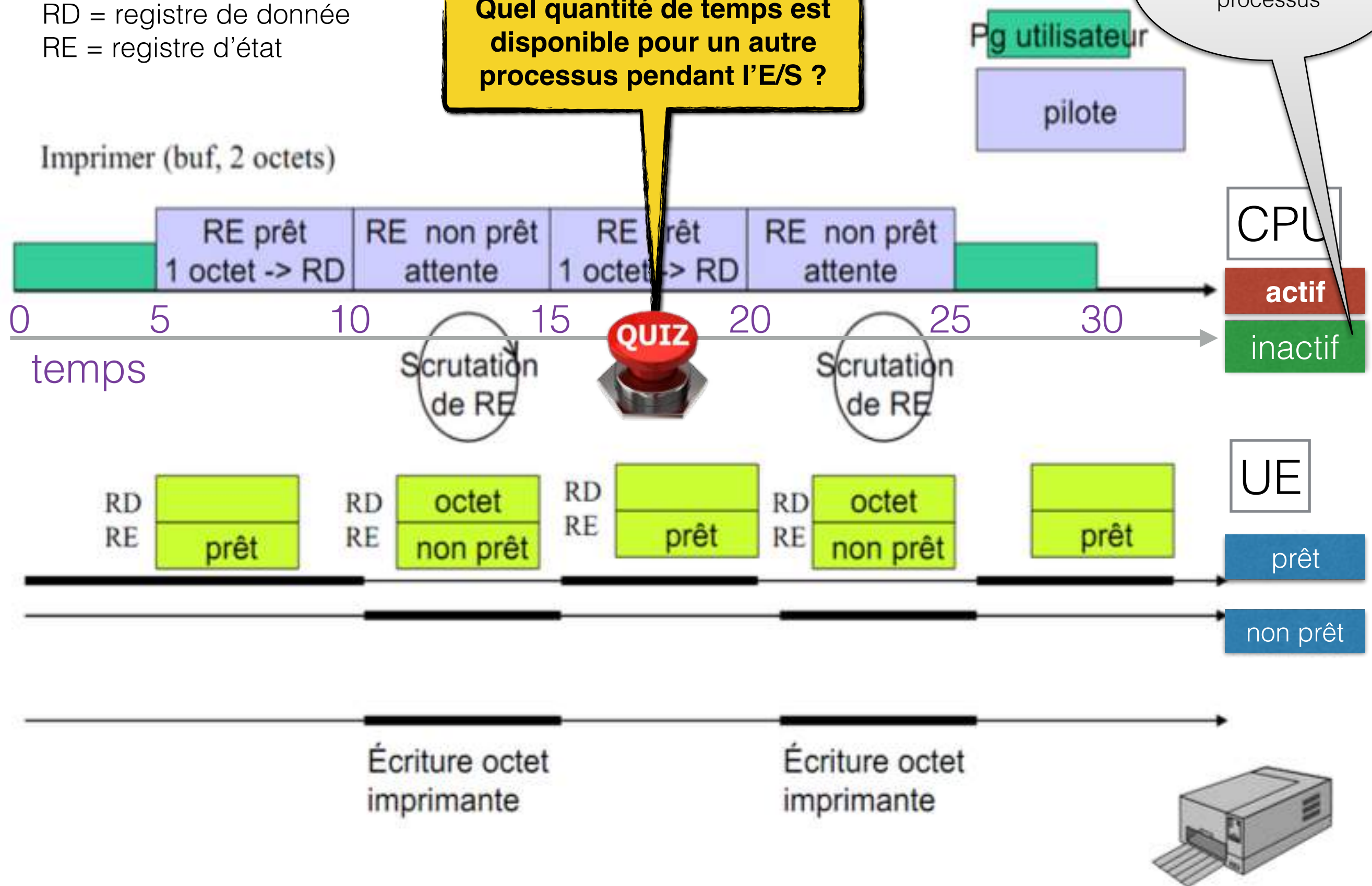
# Protocole attente active, exemple d'une imprimante

RD = registre de donnée

RE = registre d'état

**Quel quantité de temps est disponible pour un autre processus pendant l'E/S ?**

« Inactif » % E/S = disponible pour d'autres processus



## b) Protocole par interruption



**Interruption** : dispositif, incorporé au niveau du séquenceur, qui enregistre et traite les signaux d'interruption envoyés au CPU :

1. Arrêter le programme en cours ;
2. Exécuter le programme de service de l'interruption ;
3. Rétablir l'état de la machine ;
4. Reprendre l'exécution du programme interrompu.

L'unité d'échange délivre une interruption à destination du processeur lorsqu'elle est prête à délivrer ou recevoir une nouvelle donnée.

Le processeur, à la prise en compte de cette interruption, va exécuter la routine associée qui lit la donnée présente dans RD (lecture) ou met dans RD la donnée suivante à écrire.

## **prog du pilote :**

lire le RE

tant que périphérique occupé faire  
    lire le RE

lire le 1er caractère dans RD

nb\_car = nb\_car - 1

autorise\_interruption\_periph

## **gestionnaire interruption()**

interdit\_interruption\_periph

si (nb\_car > 0) faire

    lire le car suivant dans RD

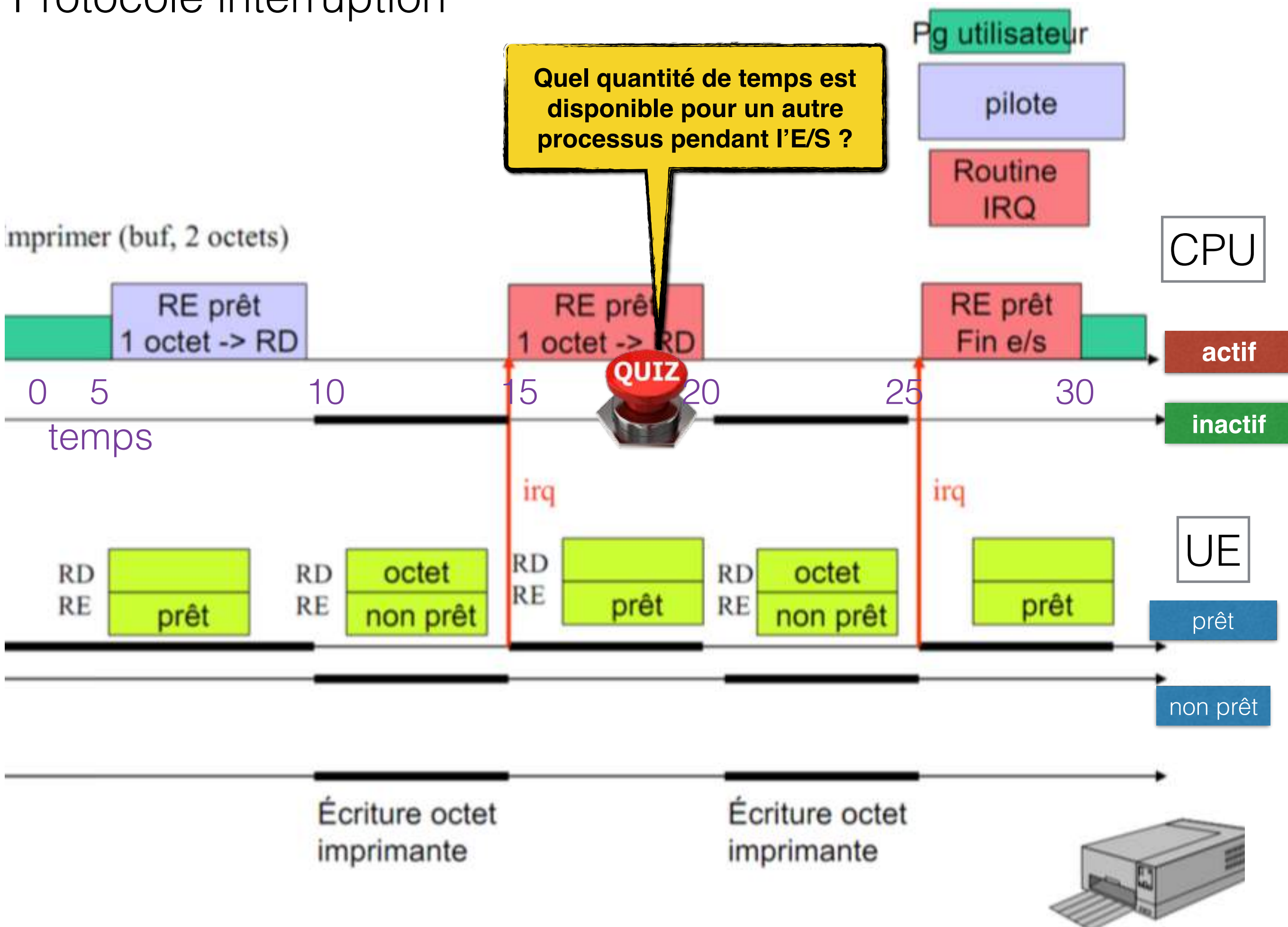
    nb\_car = nb\_car - 1

autorise\_interruption\_periph

- ➡ Le CPU n'attend pas : il exécute d'autres processus entre 2 E/S
- ➡ Cependant, il est seul à pouvoir accéder à la RAM : il a en charge les transferts RAM <-> RD périphérique
- ➡ Temps perdu à échanger les processus dans le CPU



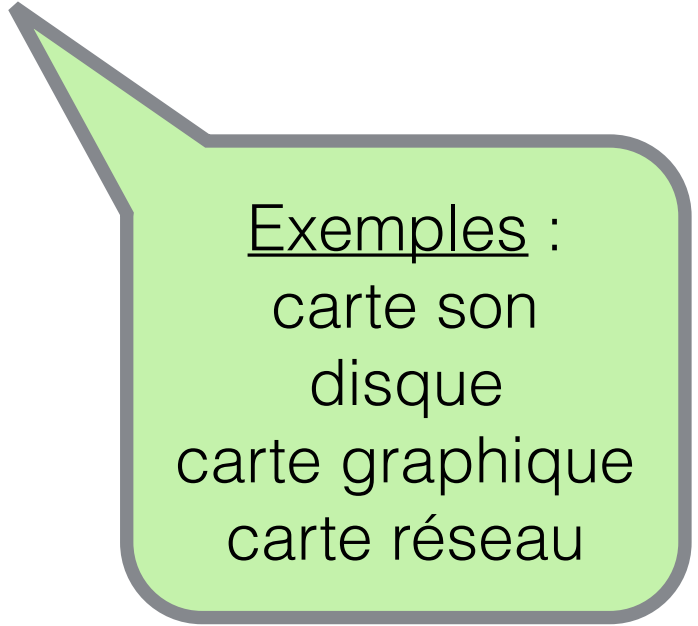
# Protocole interruption



## C) protocole Accès Direct à la Mémoire (DMA)

Le DMA contient une entité active

Le DMA permet aux UE d'accéder à la mémoire centrale sans intervention du processeur.



Exemples :  
carte son  
disque  
carte graphique  
carte réseau

❖ Les périphériques par « blocs » sont gérés par le DMA sur un PC :

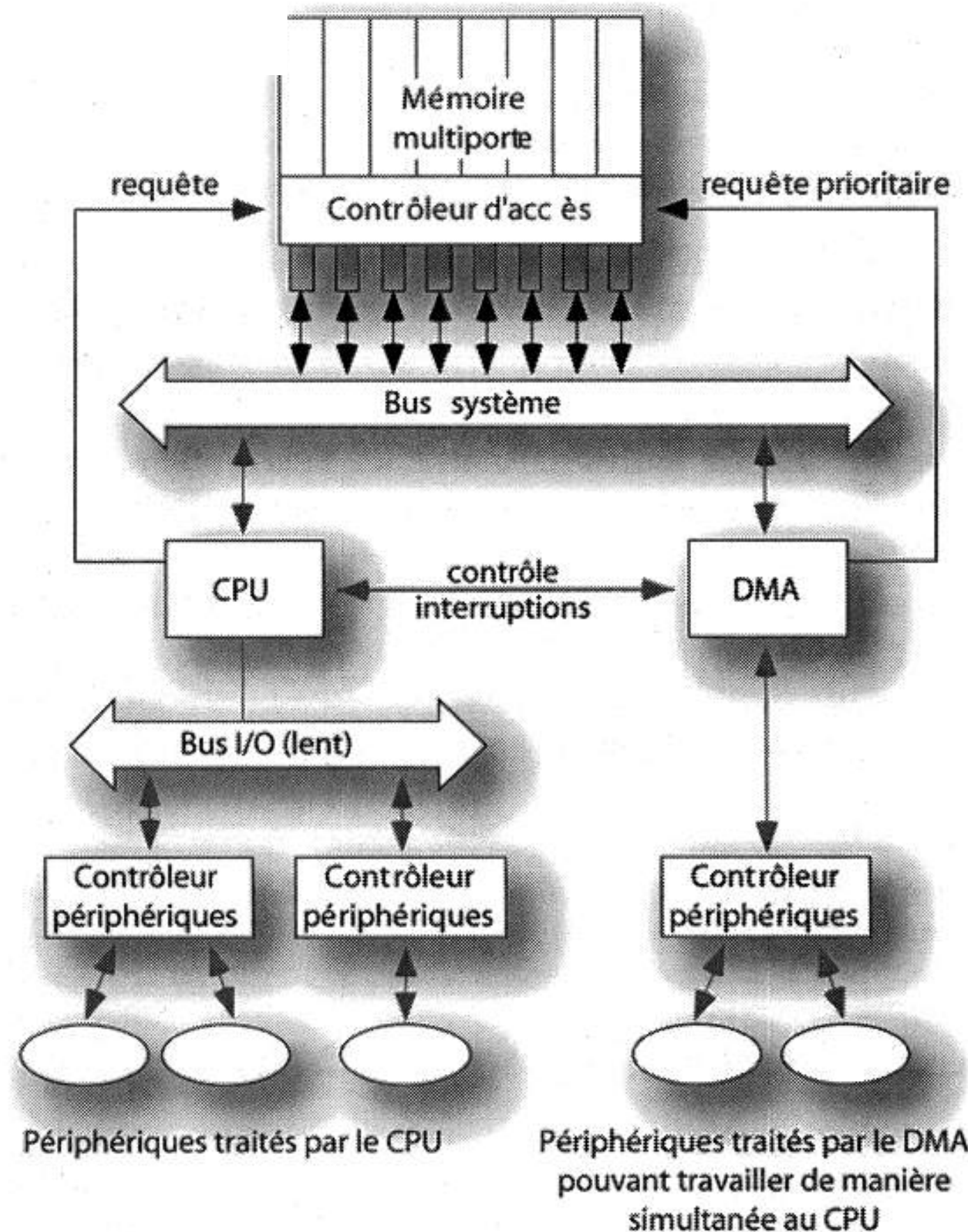
- Le contrôleur de disquettes
- Les contrôleurs de disques durs
- La carte son
- Le port parallèle (imprimante)
- etc



## C) protocole Accès Direct à la Mémoire (DMA)

Le DMA est connecté entre un contrôleur de périphérique et le bus système, permettant ainsi au périphérique d'accéder à la mémoire sans passer par le CPU.

- Transfère des blocs de données ;
- Prioritaire sur le CPU ;
- Intégrité des données non vérifiée ;
- Possède son propre RA et RM ainsi qu'un compteur (pour savoir quand signaler au CPU que les données sont prêtes).



## **prog du pilote :**

```
registre_adresse_DMA = adresse du  
    1er caractère en mémoire  
registre_cpt_DMA = nb_cars  
registre_cmd_DMA = lecture  
autorise_interruption_periph  
lancer_DMA()
```

## **gestionnaire interruption()**

```
interdit_interruption_periph  
vérifier que transfert s'est bien passé
```

- ➔ Le CPU n'a plus qu'à initialiser l'opération (remplir les registres du DMA)
- ➔ Combiner au mécanisme d'interruption, le DMA permet de décharger le processeur de la réalisation des E/S
- ➔ Lorsque l'E/S est terminée, le DMA émet une interruption pour avertir le CPU

# Ordonnancement des requêtes

- Certains périphériques « lents » et souvent demandés peuvent se retrouver avec plusieurs requêtes concurrentes
- Le pilote peut avoir alors à gérer l'ordre d'accès au périphérique pour ces requêtes
- Par exemple, pour un disque mécanique, le temps de positionnement du bras de disque est le plus coûteux en temps -> algorithmes d'ordonnancement dédiés :

**FCFS** (First Come, First Served)

**SSTF** (Shortest Seek Time First)

**SCAN** (ascenseur ou balayage)

# Exercice

- on doit servir des requêtes accédant aux pistes 50, 110, 25, 105, 12, 100, 40, 45, 10, 80, 88
- le bras est initialement sur la piste 90 (sur 150 disponibles)

FCFS

First Come,  
First Served



Dans quel ordre seront traité ces requêtes ?  
Quel est le total des déplacements réalisés ?

# Exercice

- on doit servir des requêtes accédant aux pistes 50, 110, 25, 105, 12, 100, 40, 45, 10, 80, 88 avec un bras initialement sur la piste 90 (sur 150 disponibles)

SSTF

Shortest Seek  
Time First



Dans quel ordre seront traité ces requêtes ?  
Quel est le total des déplacements réalisés ?



# Exercice

- on doit servir des requêtes accédant aux pistes 50, 110, 25, 105, 12, 100, 40, 45, 10, 80, 88 avec un bras initialement sur la piste 90 (sur 150 disponibles)

SCAN

balayage  
gche<->dte



Dans quel ordre seront traité ces requêtes ?  
Quel est le total des déplacements réalisés ?

# Le monde Linux



# PÉRIPHÉRIQUES ENTRÉES/SORTIES

- ❖ Selon le type le format de données, on distingue aussi deux autres catégories de périphériques :
- ❖ **Périphériques par caractères** dans lesquels, on accède à l'information caractère par caractère (exemple : le clavier) mais on ne peut spécifier une adresse ni rechercher une information : on reçoit ou on envoie simplement un flux de caractères.
- ❖ **Périphériques par blocs** dans lesquels, on ne peut accéder à l'information que par blocs et chaque bloc possède une adresse (exemple : le disque).

# Linux gère les E/S comme des fichiers spéciaux

- Exemple : une imprimante est le fichier /dev/lp0, attaché au pilote de l'imprimante
- Chaque fichier spécial a 3 attributs :
  - son type (b ou c)
  - un numéro majeur (1 à 255), identifiant le pilote gérant ce périphérique
  - un numéro mineur, identifiant pour son pilote le numéro de périphérique physique (un pilote peut gérer plusieurs périph. de même type)

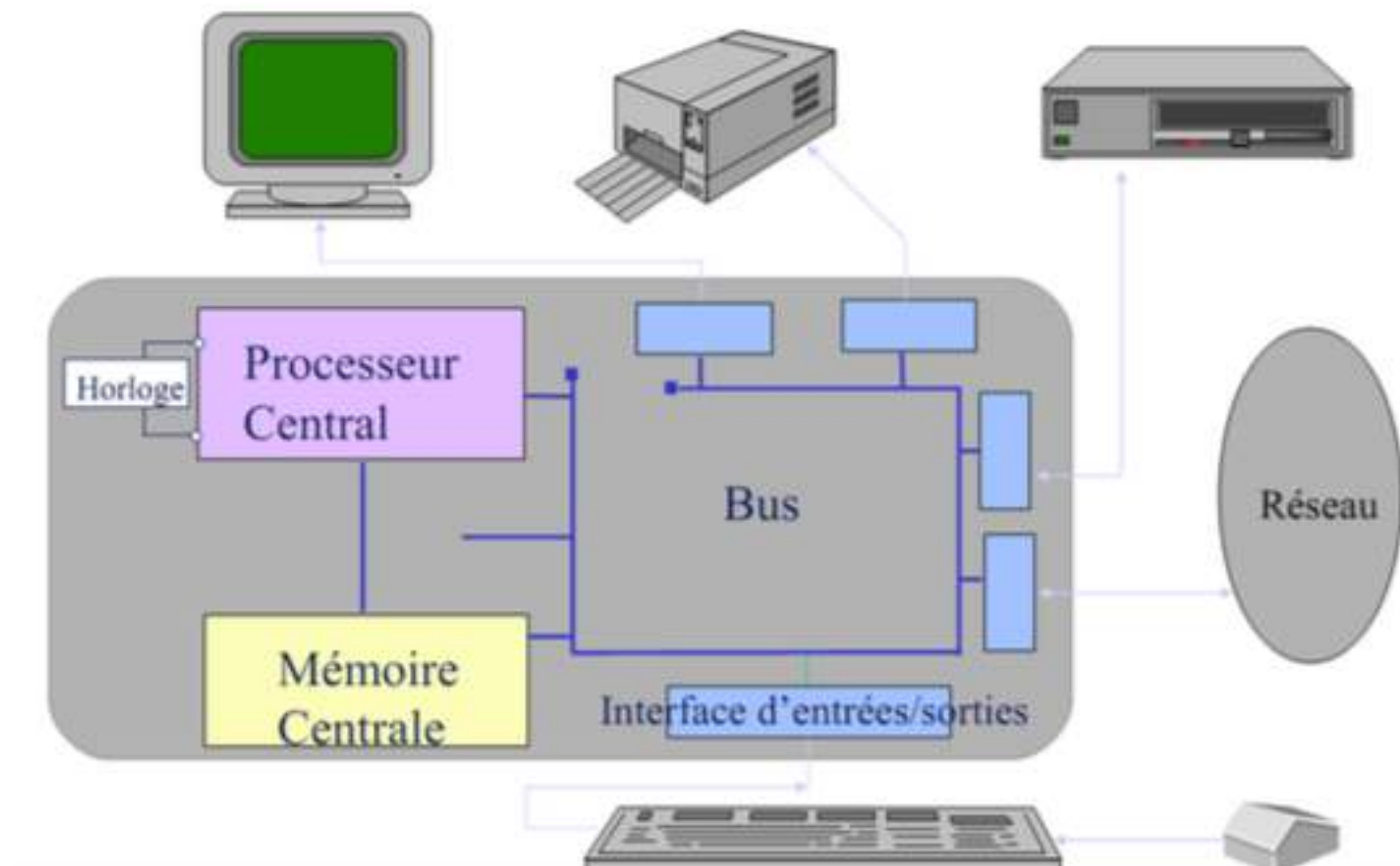
Exemple : `ls -l /dev`

# Les BUS



# BUS

- Les transferts d'information entre le processeur, la mémoire et les différents contrôleurs d'E/S s'effectuent à travers des dispositifs de communication appelés bus
- Un **bus** est un ensemble de liaisons physiques (câbles, circuits imprimés) reliant différents composants de l'ordinateur et leur permettant de communiquer
- Maximum 2 unités peuvent utiliser un bus en même temps ;



❖ On peut classer les éléments connectés à un bus en deux catégories:

➤ Les **esclaves:** sont passifs et répondent à des requêtes (e.g. mémoire)

➤ Les **maîtres:** sont actifs et initient des requêtes. Ils prennent le contrôle du bus (e.g. processeur)

❖ Il ne peut pas y avoir 2 maîtres actifs simultanément ⇒

Besoin d'arbitrage pour l'accès au bus.

# Quizz time



- Dans quelle situation aujourd'hui nous avons vu que plusieurs maîtres pouvaient essayer de dialoguer sur le même bus ?
- De quel bus s'agit-il ?

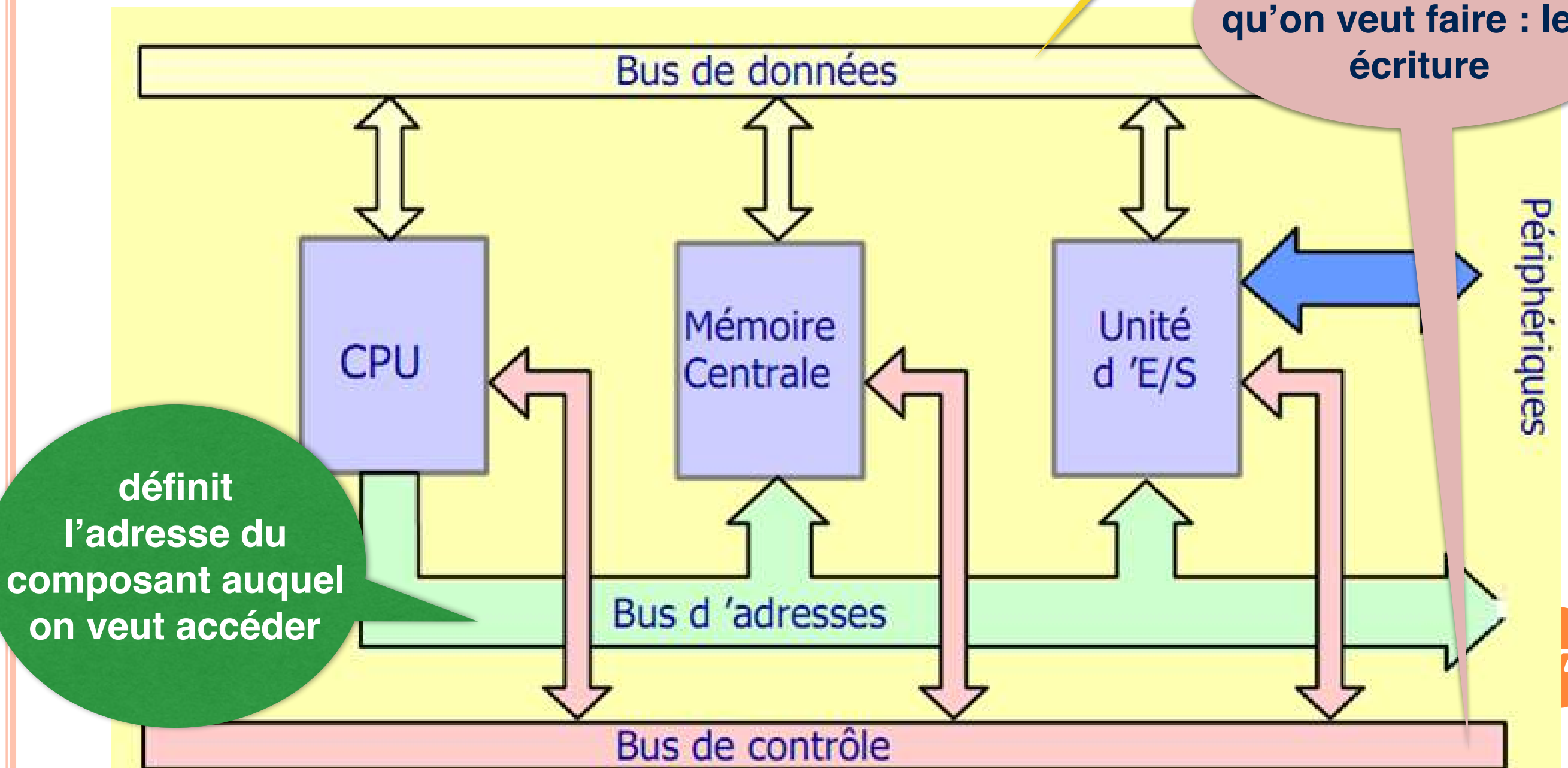


# DÉCOMPOSITION

les données  
échangées

indique opération  
qu'on veut faire : lect/  
écriture

- ❖ Un bus est capable de véhiculer des signaux correspondant fondamentalement à trois types d'informations : adresses; données et commandes (ou contrôle).

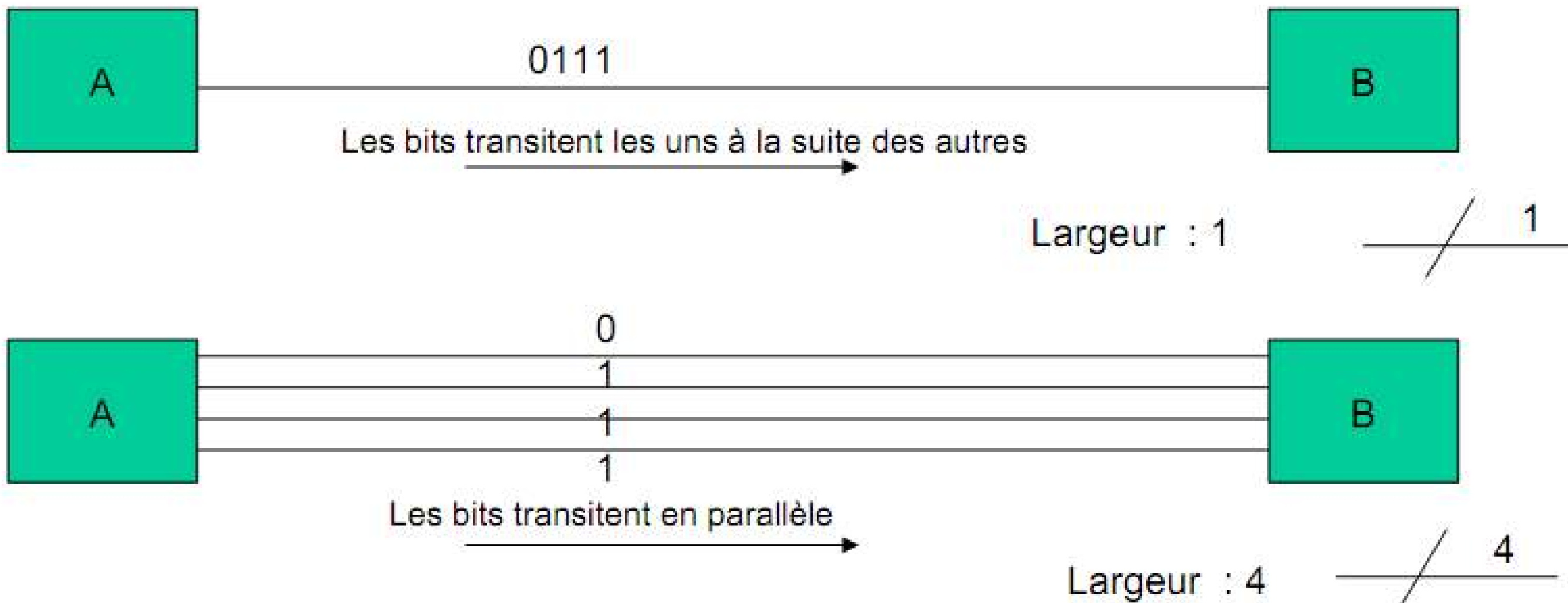




# CARACTÉRISTIQUES

❖ **Largeur du bus** : nombre de bits que le bus peut véhiculer simultanément

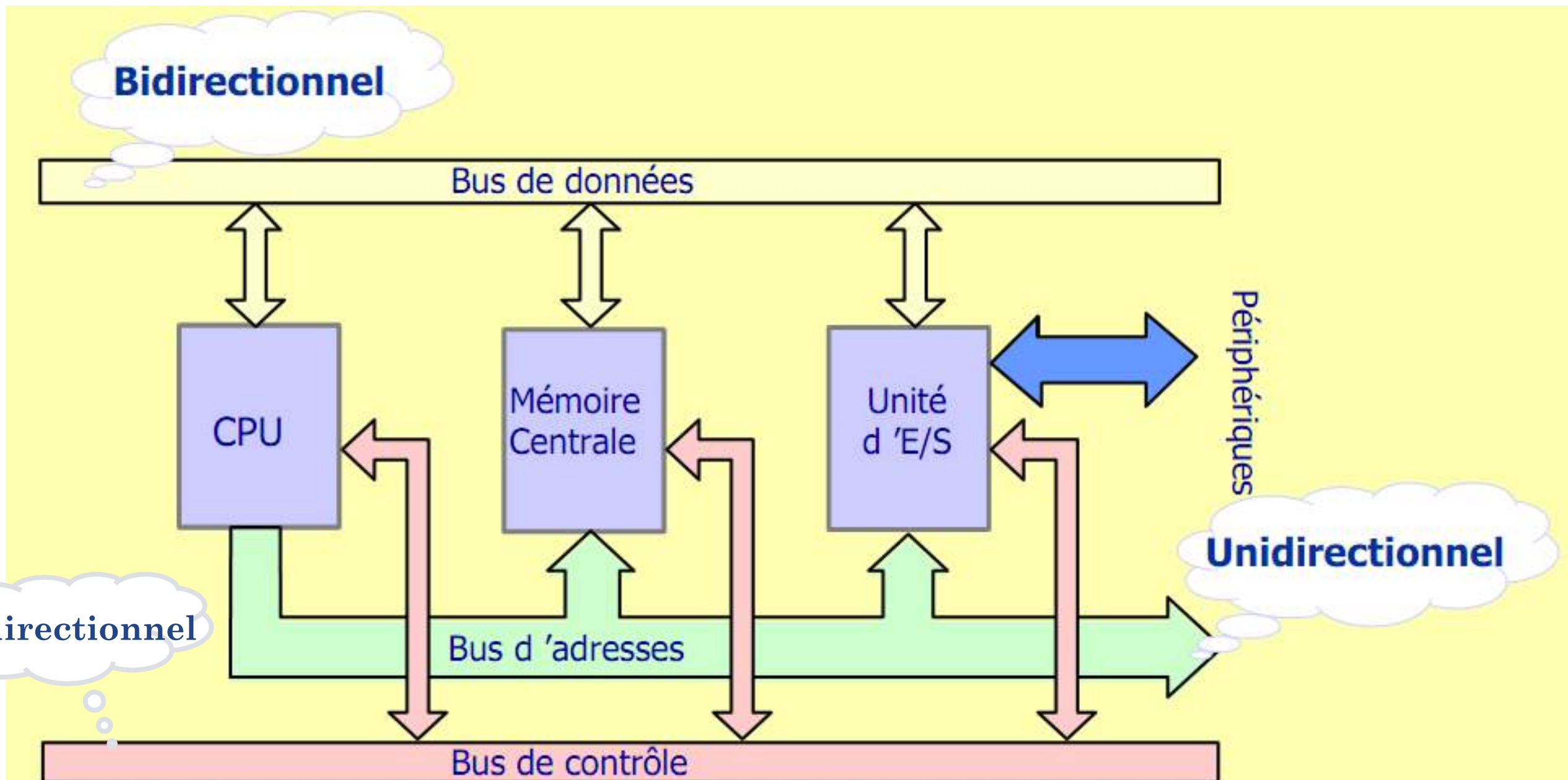
➤ *Bus série / Bus parallèle*



# CARACTÉRISTIQUES

## ❖ Direction de transmission

- *Simplex – unidirectionnel*
- *Duplex – bidirectionnel*



# CARACTÉRISTIQUES

## ❖ Cadencement du bus

### ➤ *Bus synchrones / Bus asynchrones :*

- Bus synchrone dispose d'un horloge propre qui définit le cycle du bus.
- Bus asynchrone n'y a pas d'horloge associée au bus, les dispositifs dialoguent par le biais de signaux de disponibilité

## ❖ **Débit du bus :** nombre d'octets véhiculés par seconde exprimés en Mo/s ou Mb/s

*Débit (Bus synchrone unidirectionnel) = largeur \* fréquence*

*où la fréquence = nombre de cycles par secondes (Méga-Hertz)*

# Quizz time



Dans le cas de bus synchrones unidirectionnels

- Quel est le bus avec le plus fort débit entre
  - bus PCI avec 32 bits cadencé à 133MHz et
  - bus ISA de largeur 16 bits cadencé à 133MHz ?
- Quel est le débit maximal d'un tel bus ISA ?

# Protocole pour un bus

- le bus est partagé entre différents composants
- une transaction bus se fait en deux étapes : d'abord arbitrage entre demandeurs potentiels, puis transfert des données
- L'ensemble des règles d'arbitrage et de transfert est appelé **protocole de gestion du bus**


# Protocole pour un bus

Quelques détails du protocole :

- 1.obtention du contrôle par un maître
- 2 envoi d'une adresse qui identifie l'esclave
- 3.envoi d'une requête (lecture, écriture, ...)
- 4.envoi/réception de données vers/de l'esclave
- 5.libération du bus

# Le cas du bus I<sup>2</sup>C

(présent sur Raspberry)



l'interface  
avec le shield  
Grove passe par  
ce bus

<http://innovelectronique.fr/2013/03/02/utilisation-du-bus-i2c-sur-raspberrypi/>





# Présentation



## **I2C : *Inter Integrated Circuit***

développé au début des années 80 par Philips Semiconductor pour permettre de relier facilement à un microprocesseur les différents circuits d'un téléviseur moderne.



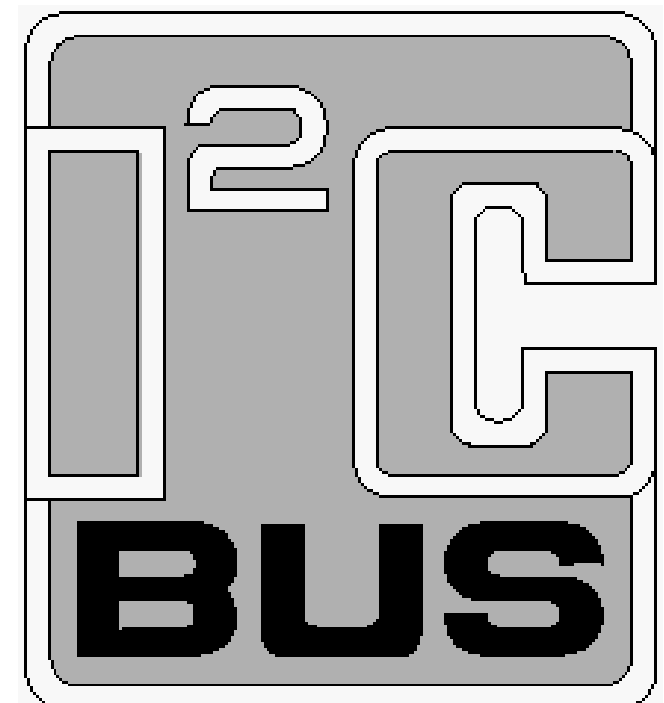
## **But :**

faire communiquer entre eux des composants électronique très divers grâce à seulement 3 fils :

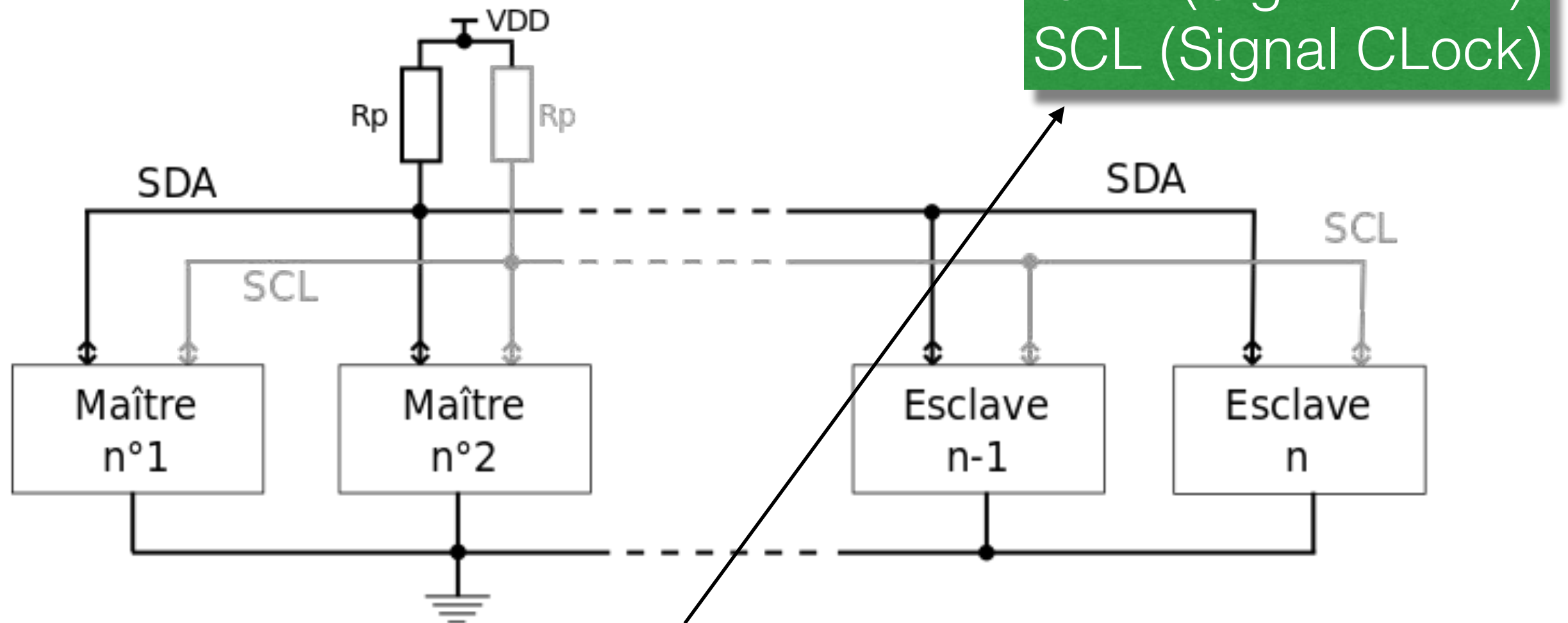
- Signal de donnée : SDA
- Signal d'horloge : SCL
- Signal de référence électrique : masse



**Un bus I2C est-il série ou parallèle ?  
Est-il synchrone ou asynchrone ?**









# Bus I<sup>2</sup>C



- I<sup>2</sup>C est un bus **série** synchrone duplex.
- Plusieurs équipements, soit maîtres, soit esclaves, peuvent être connectés au bus.
- Les échanges ont toujours lieu entre **un seul maître** et un (ou tous les) esclave(s)



# Caractéristiques du bus I2C

-  Deux lignes uniquement (SDA et SCL) + masse
-  1 adresse unique pour chaque périphérique
-  Bus multi-maître, détection des collisions et arbitrage
-  Bus série, 8 bits, bi-directionnel à 100 kbps (*standard mode*), 400 kbps (*fast mode*), 3,2 Mbps (*high-speed mode*)
-  Filtrage intégré : réjection des pics parasites
-  Nombre de circuits uniquement limité par la capacitance maximale du bus : 400 pF

Une communication **SCL** est générée uniquement par un **maître**, tandis que sur SDA, le maître et l'esclave peuvent parler



# Prise de contrôle du bus



Le bus doit être au repos avant la prise de contrôle SDA et SCL à 1



Pour transmettre des données, il faut surveiller :

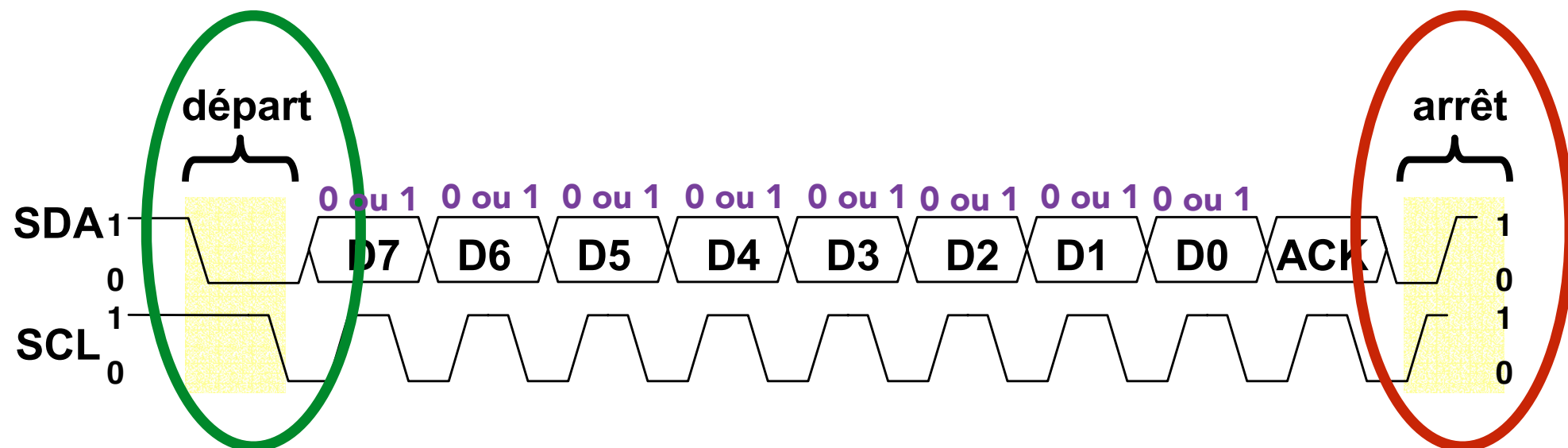
- La condition de départ : SDA passe à 0, SCL reste à 1
- La condition d'arrêt : SDA passe à 1, SCL reste à 1

START

STOP



Après avoir vérifié que le bus est libre, puis pris le contrôle de celui-ci, le circuit en devient le maître : c'est lui qui génère le signal d'horloge

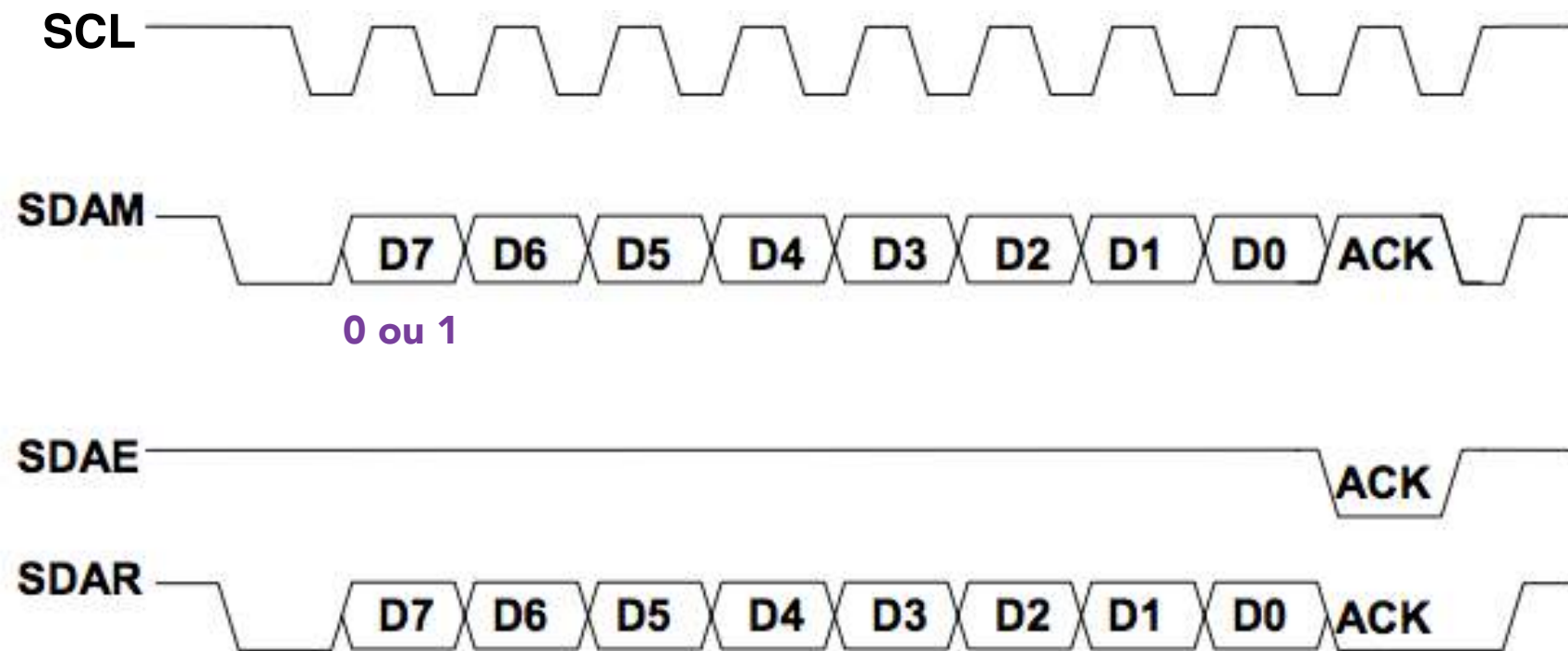


# Transmission d'un octet

- Le maître transmet le bit de poids fort D7 sur SDA

Il valide la donnée en appliquant un niveau '1' sur SCL

- Lorsque SCL retombe à '0', il poursuit avec D6, etc. jusqu'à ce que l'octet complet soit transmis



SCL : Horloge imposée par le maître  
SDAM : Niveaux de SDA imposés par le maître  
SDAE : Niveaux de SDA imposés par l'esclave  
SDAR : Niveaux de SDA réels résultants

- Il envoie le bit ACK à 1 en scrutant l'état réel de SDA (ACK ou NACK positionné par l'esclave)
- L'esclave impose un niveau 0 sur SDA pour signaler qu'il a reçu le bit : ACK (*acknowledge*)
- Le maître voit SDA à 0 et peut passer à la suite



# Transmission d'un octet

L'acquittement :  
**ACKNOWLEDGE**

Voyons ça de près



Le maître libère la ligne SDA



L'esclave force la ligne SDA  
au niveau bas (trait gras)



Le maître envoie une  
impulsion sur l'horloge SCL



Lorsque l'impulsion retombe à  
zéro, l'esclave libère SDA



# Transmission d'une adresse



Nombre de composants important : nécessité de définir pour chacun une adresse unique



Adresse codée sur 7 bits, définie par :

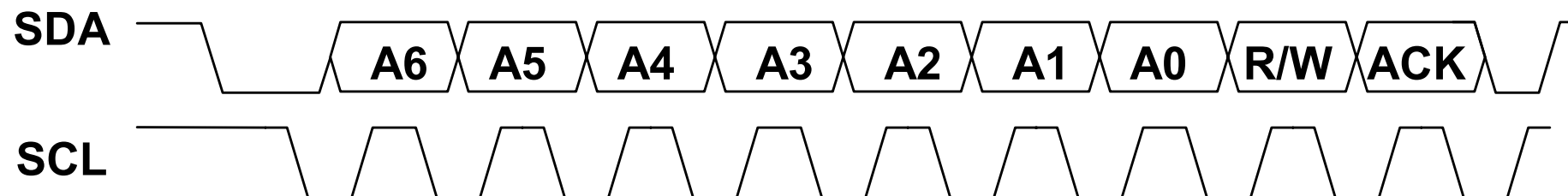
- Son type
- L'état appliqué à un certain nombre de ces broches



Adresse transmise sous la forme d'un octet au format particulier :

- D7 à D1 : 7 bits d'adresse A6 à A0
- D0 : bit R/W qui détermine si le maître veut lire ou écrire

0 écriture,  
1 lecture



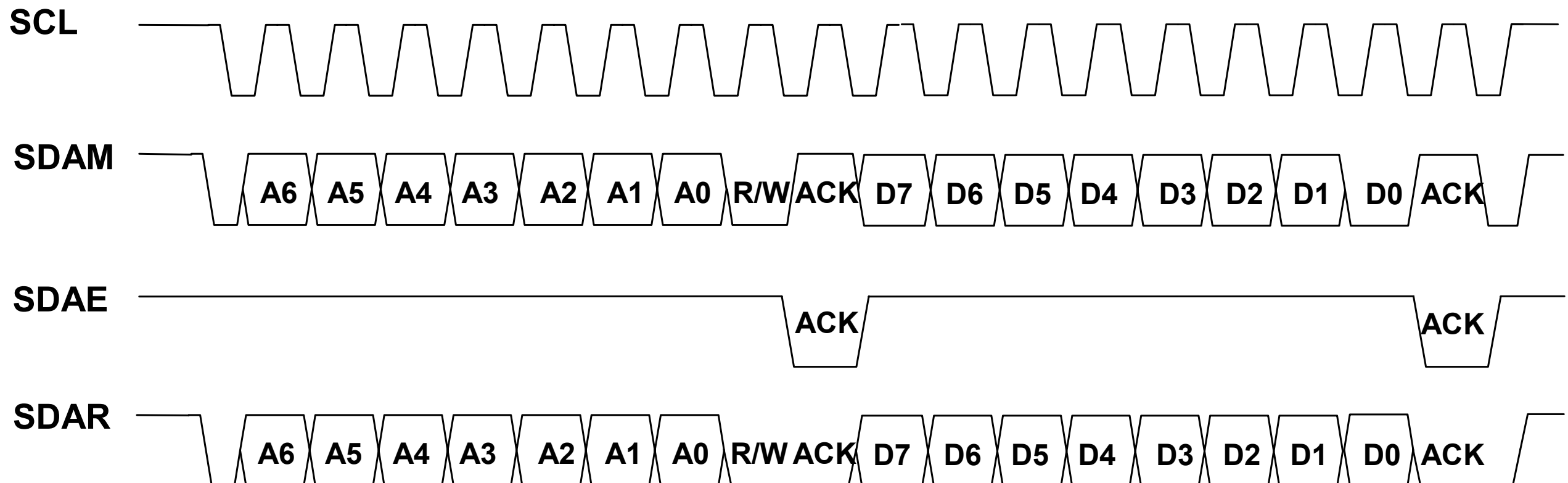




# Écriture d'une donnée

1. Envoi de l'adresse de destination
2. Sélection du mode écriture (R/W à 0)
3. Envoi de la donnée

**Note :** il peut être nécessaire d'attendre ACK avant de poursuivre (écriture dans des mémoire, etc.)





# Exercice

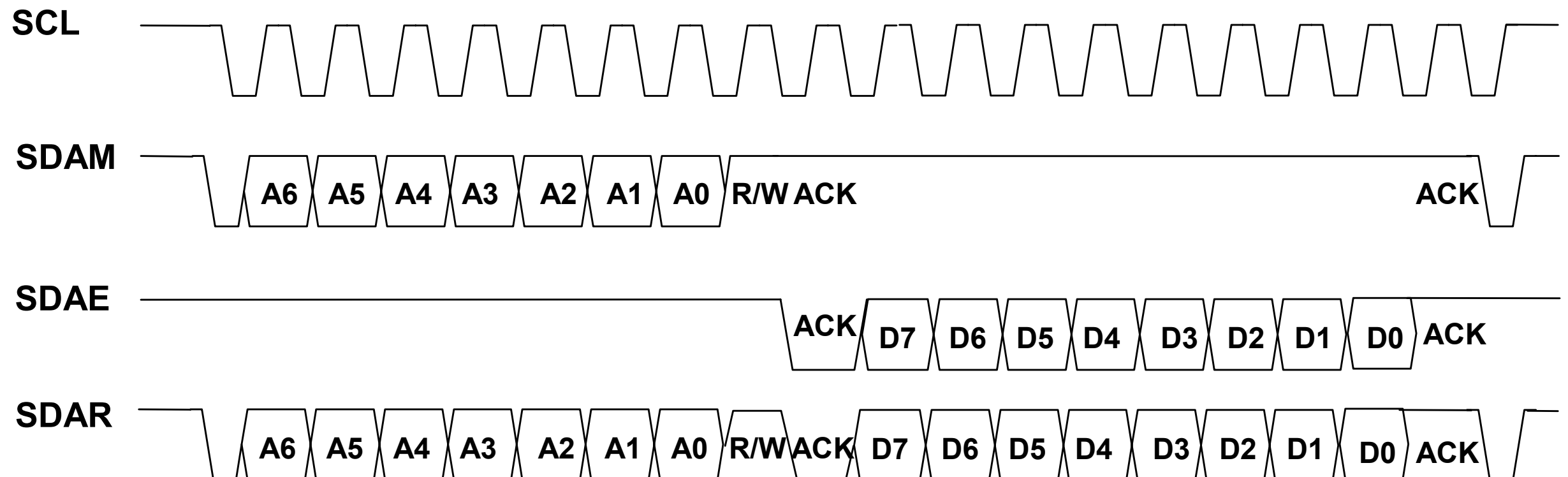
- Qu'est-ce qu'une RTC pour raspberry ?
  - Quel est l'intérêt d'un tel élément ?
  - Le circuit Dallas DS1307 est une RTC, accessible à l'adresse 1101000 (adr sur 7 bits). Pour positionner la date (valeur de 1 à 31), il faut la l'envoyer dans le registre 04h de ce circuit.
1. Listez les données qui circulent dans l'ordre où elles vont circuler sur le bus
  2. Dessinez la liste de bits envoyés sur le bus I2C pour fixer la date à 27 dans le RTC (on envoie au RTC d'abord l'adresse où écrire, puis la valeur) :
  3. Dessinez les SDA (SDAM, SDAE, SDAR) :

- 1)
- 2)
- 3)
- 4) .
- 5)
- 6)
- 7)
- 8)



# Lecture d'une donnée

1. Le maître envoie l'adresse puis attend l'ACK
2. L'ACK est positionné par l'esclave, puis celui-ci émet les données sur SDA
3. Ensuite, le maître positionne ACK à '0' pour continuer la lecture, ou '1' pour stopper la transmission





- Les registres d'adresses 00h à 06h du DS1307 contiennent respectivement les secondes, minutes, heures, jours, dates, mois et années. Décrivez le comportement du SDA à la lecture, d'une seule traite, du contenu des registres d'adresses 00h à 06h du DS1307 :

# →I2C Protocol Summary

49

START	HIGH to LOW transition on SDA while SCL is HIGH
STOP	LOW to HIGH transition on SDA while SCL is HIGH
DATA	<p>8-bit word, MSB first (Address, Control, Data):</p> <ul style="list-style-type: none"><li>- Must be stable when SCL is HIGH</li><li>- Can change only when SCL is LOW</li><li>- Number of bytes transmitted is unrestricted</li></ul>
ACKNOWLEDGE	<ul style="list-style-type: none"><li>- Done on each 9th clock pulse during the HIGH period</li><li>- The transmitter releases the bus - SDA goes HIGH</li><li>- The receiver pulls DOWN the bus line - SDA goes LOW</li></ul>
CLOCK	<ul style="list-style-type: none"><li>- Generated by the Master(s)</li><li>- Maximum speed: (100, 400, 1000, 3400 kHz) but NO min</li><li>- A receiver can hold SCL low when performing another function (transmitter in a Wait state)</li><li>- A master can slow down the clock for slow devices</li></ul>
ARBITRATION	<ul style="list-style-type: none"><li>- Master can start a transfer only if the bus is free</li><li>- Several masters can start a transfer at the same time</li><li>- Arbitration is done on SDA line</li><li>- Master that lost the arbitration must stop sending data</li></ul>

# Sources

- ✻ Livre de J. Delacroix (Dunod, 28€)
- ✻ Cours de Clément Jonquet
- ✻ Cours de S. AROUSSI
- ✻ Cours de C. Diou
- ✻ Wikipedia



# TP sur le bus I<sup>2</sup>C sur Raspberry

<http://innovelectronique.fr/2013/03/02/utilisation-du-bus-i2c-sur-raspberrypi/>

**Questions** : aidez-vous de la page ci-dessus et d'autres pour :

- installez les outils I2C (*I2C tools*) sur le raspberry
- déterminer à quelle vitesse fonctionne le bus I2C du raspberry
- savoir à quel fichier spécial il est affecté par Linux
- détecter quelle adresse est assignée au shield Grove sur ce bus
- Envoyer par ligne de commande Unix un signal d'allumage sur une LED connectée au shield

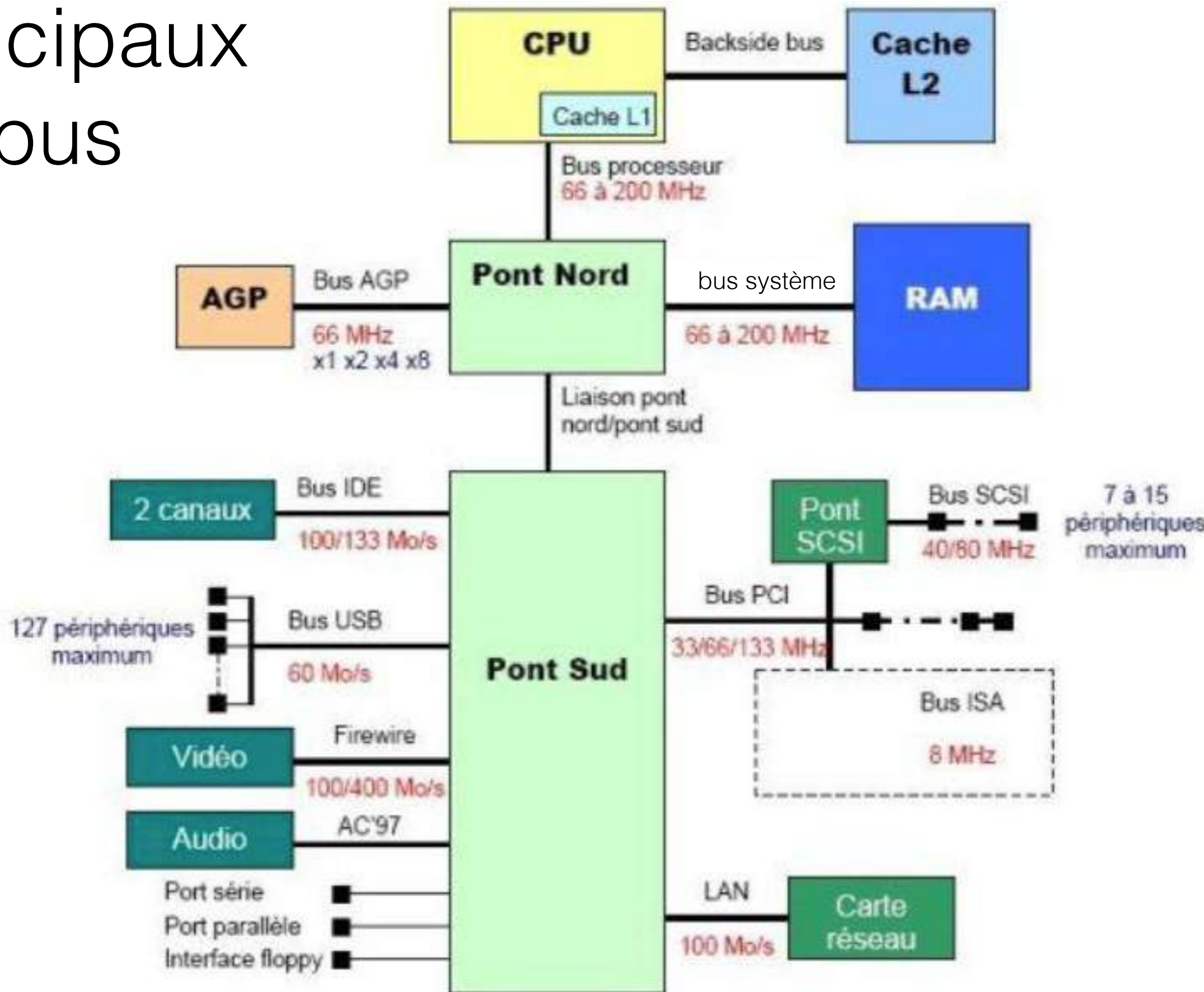
## Notes pour le TP :

- aptitude est un gestionnaire de paquets plus récents/performant que apt-get
- si ce n'est déjà fait, il faut permettre l'utilisation de l'interface i2c sur votre raspberry : Raspberry pi configuration -> Interfaces -> cocher I2C enabled puis rebootez
- i2cdetect -y 1 (si /dev/i2c-1 est apparu et non 0)
- i2c-bcm2708 est le pilote bas niveau. i2c-dev crée les entrées /dev/i2c bien pratiques
- SMBus (System Management Bus) is a subset from the I2C protocol
- When writing a driver for an I2C device try to use the **SMBus** commands if possible (if the device uses only that subset of the I2C protocol) as it makes it possible to use the device driver on both SMBus adapters and I2C adapters.

# Annexe

Information supplémentaire  
(toujours pour le même prix)

# Principaux bus



# Bus I2C : un cas particulier pour la transmission d'adresses

- Cas particulier : les **mémoires** accessibles par I2C
  - Espace adressable plus grand que les autres circuits :  
adresses codées sur 2 octets ou plus :
    - Premier octet = adresse du circuit auquel on s'adresse
    - Octet suivant = adresse interne de la mémoire
- Adresses réservées
  - 00000XXX et 111111XX réservées à des modes de fonctionnement particuliers