Systèmes embarqués

Introduction à la programmation micro-contrôleur

Julien DeAntoni

Merci à Jean-Philippe Babau pour m'avoir permi la réutilisation d'une partie de ses supports

Université Nice SOPHIA ANTIPOLIS
Polytech'Nice-Sophia
INRIA SOPHIA ANTIPOLIS
i3S

Pourquoi ce cours ?

- Différents types de systèmes embarqués
  - Téléphones / PDA / Différents gadgets / Modem
  - Avionique / Automobile / Chaîne de production
  - Cafetière, aquarium, etc

- Différents besoins
  - Pas d'OS
  - OS temps réel dédiés
  - Linux embarqués ou pseudo Linux

Pourquoi ce cours ?

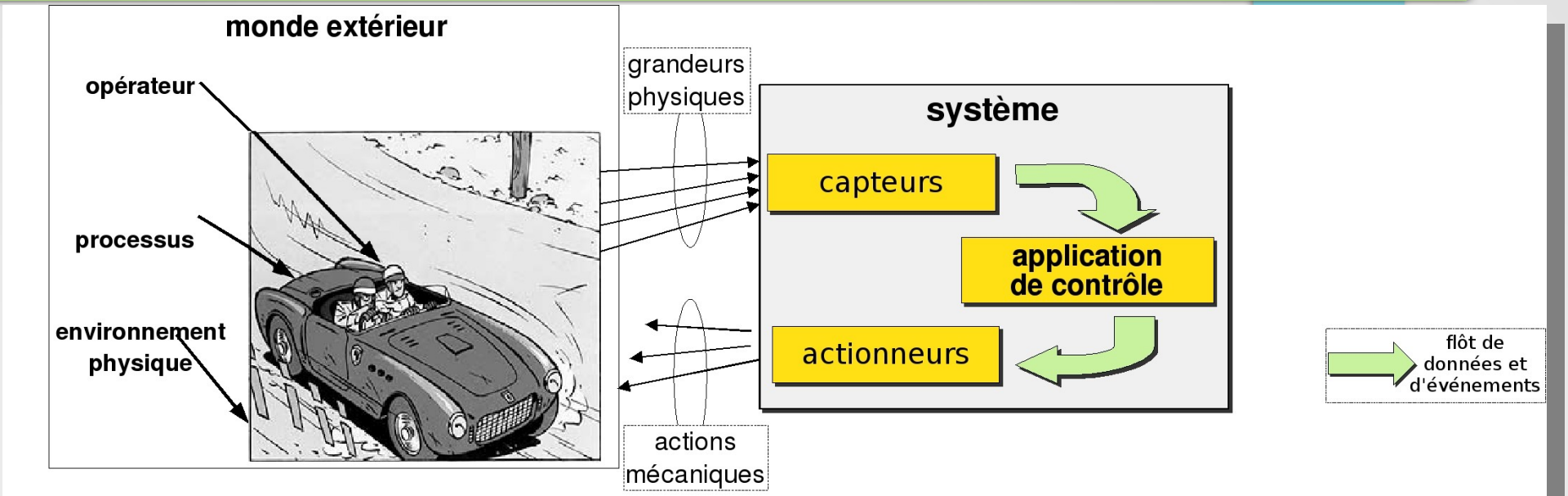
- Différents types de systèmes embarqués
  - Téléphones / PDA / Différents gadgets / Modem
  - **Avionique / Automobile / Chaîne de production**
  - **Cafetière, aquarium, etc**

- Différents besoins
  - **Pas d'OS**
  - **OS temps réel dédiés**
  - Linux embarqués ou pseudo Linux

Contenu du cours

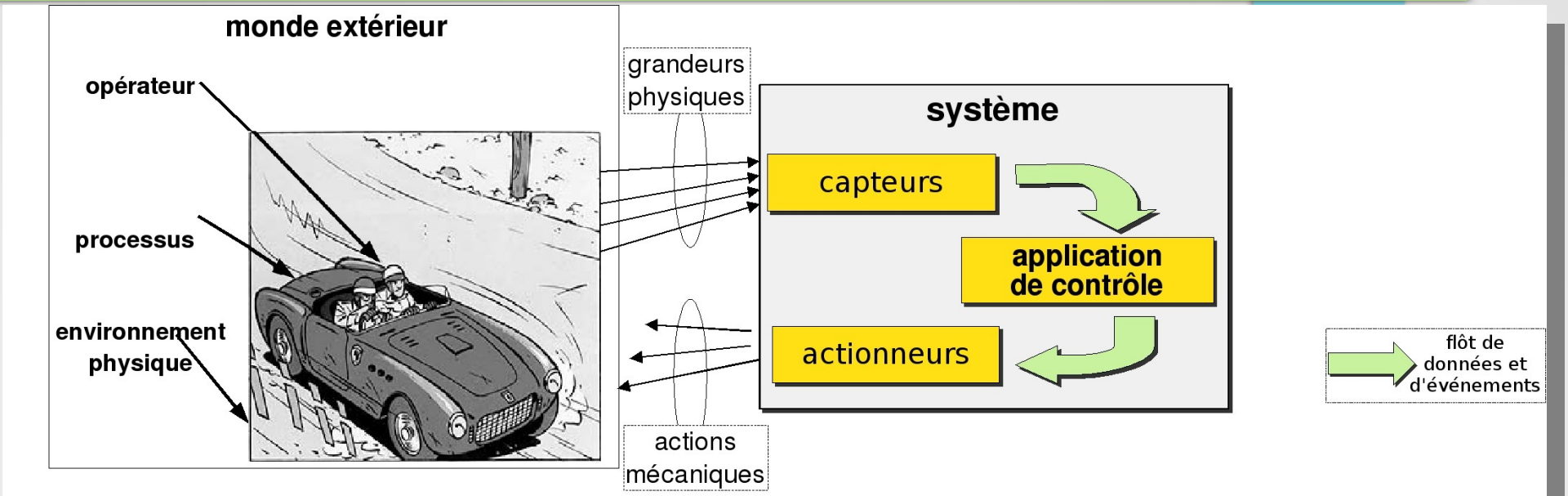
- Généralités
  - Les sytèmes considérés
  - Le développement de tels systèmes
- Programmation sans OS
  - micro-contrôleur sans OS, pourquoi, comment ?
  - Stratégie d'implémentation
    - programmation sans IT (Synchrone)
    - programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Les systèmes embarqués considérés



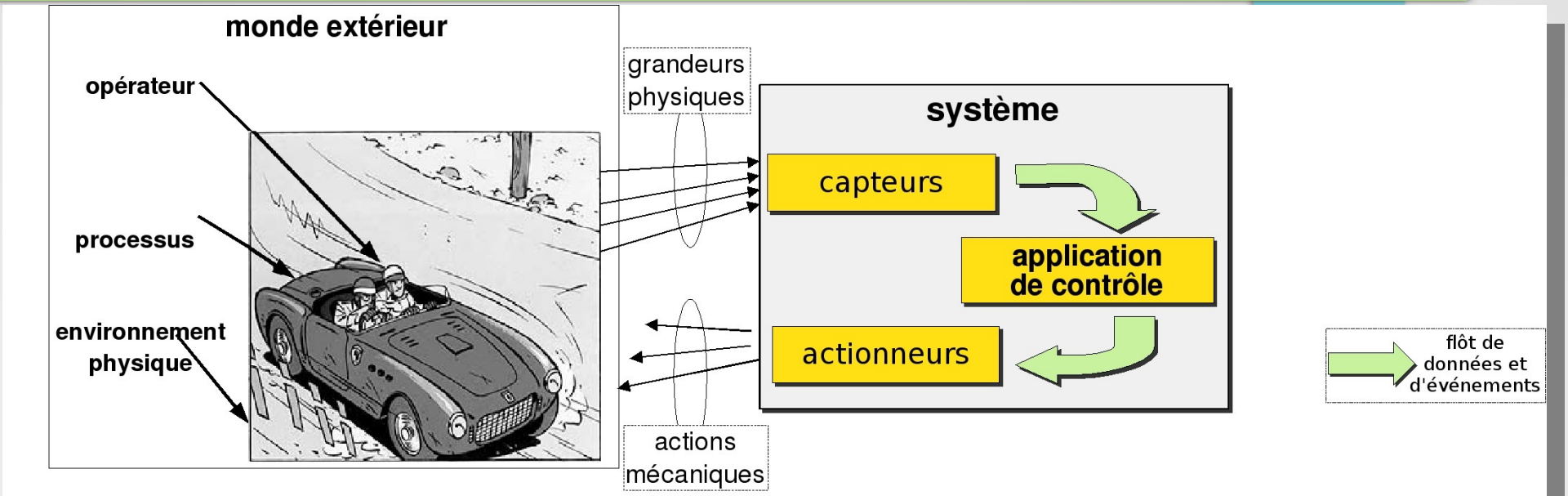
- Systèmes temps réels
  - En charge du contrôle d'un processus
  - Liés à la dynamique du processus à contrôler
  - Soumis à des contraintes temporelles

Les systèmes embarqués considérés



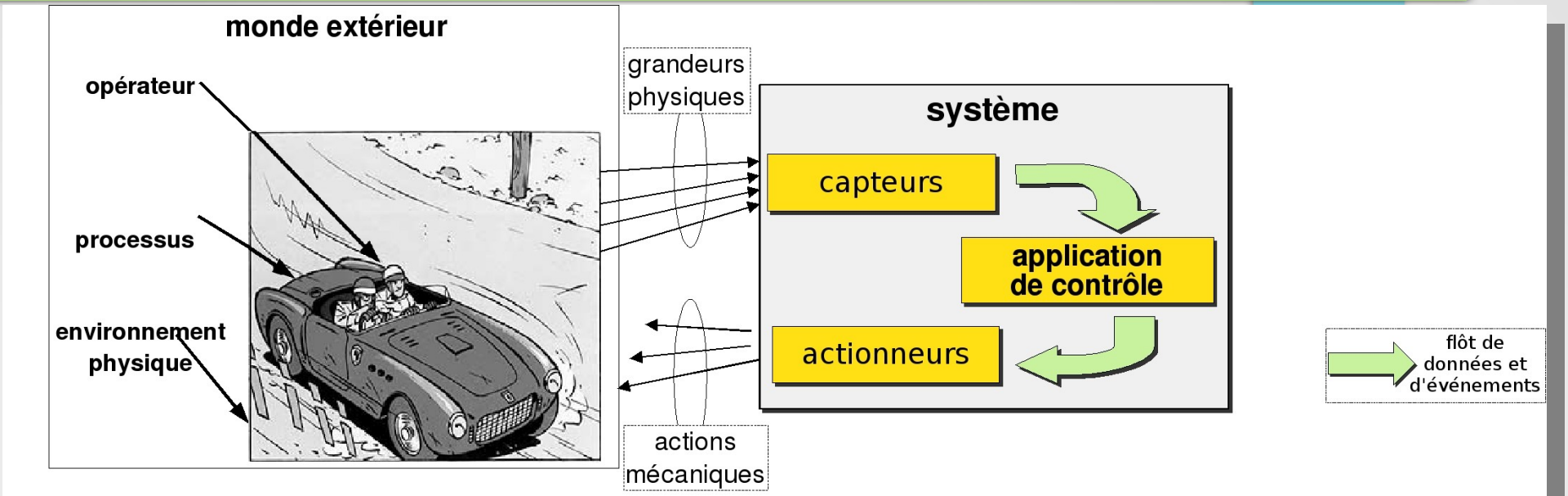
- Systèmes temps réels
  - Pas forcément rapides (contrairement aux idées reçues)
  - Prédictibles
  - (Souvent) Fortement enfouis

Les systèmes embarqués considérés



- Systèmes temps réels
  - Contraintes matérielles fortes
    - Mémoire
    - Coût
    - Taille
    - Consommation

Les systèmes embarqués considérés

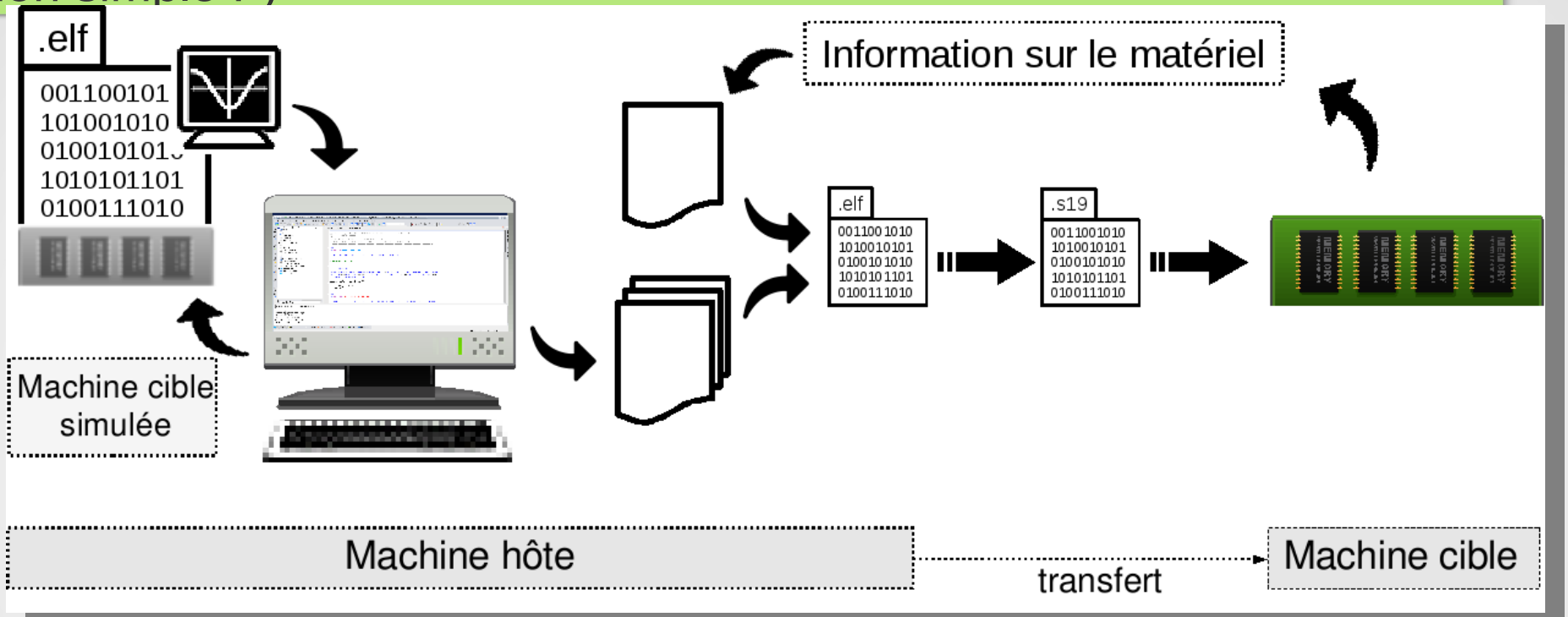


- Systèmes Critiques
  - Une faille peut mettre des vies en danger
    - Régulateur de vitesse
    - ABS
  - Prédictibilité accrue (déterminisme)

Contenu du cours

- Généralités
  - Les sytèmes considérés
  - **Le développement de systèmes temps réel**
- Programmation sans OS
  - micro-contrôleur sans OS, pourquoi, comment ?
  - Stratégie d'implémentation
    - programmation sans IT (Synchrone)
    - programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Le développement de systèmes temps réel
version simple :-)

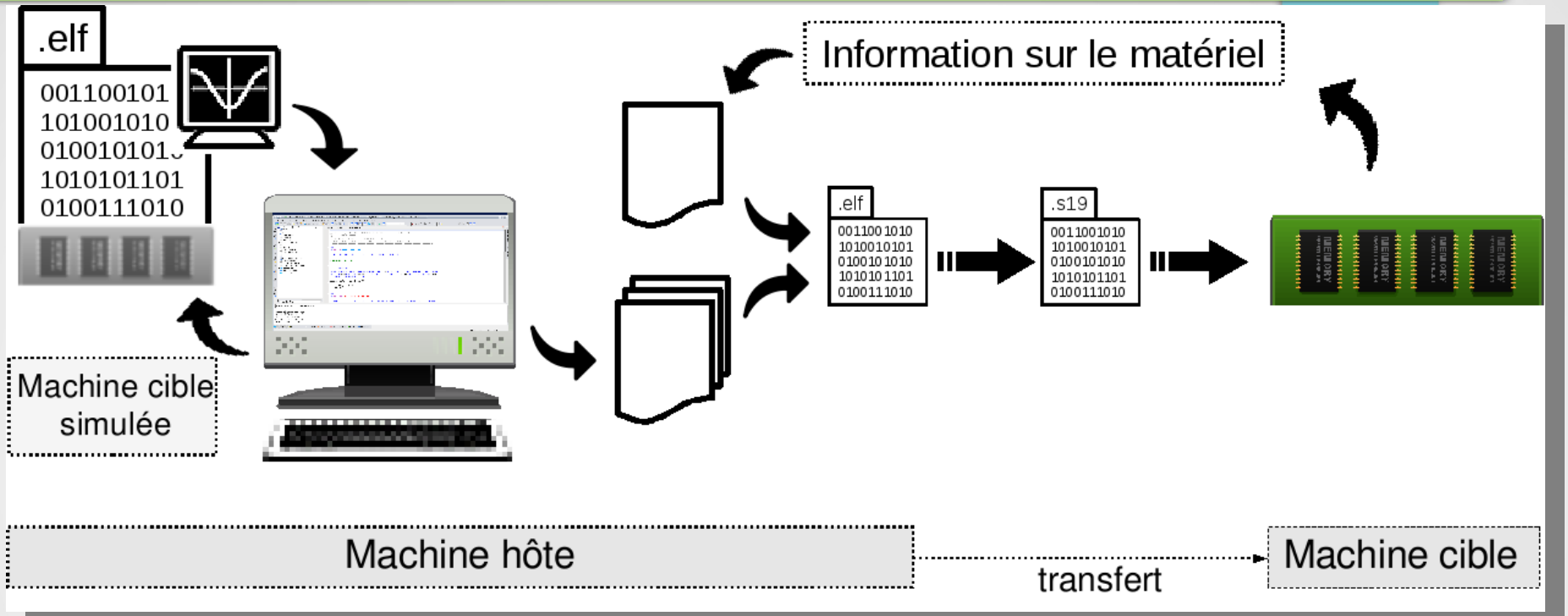


- **Différence forte machine hôte / machine cible**

  Machine hôte :

  - Entrées / Sortie Standard
  - IDE (spécifiques ou pas)
  - Simulation de la cible / environnement / processus

Le développement de systèmes temps réel
version simple :-)

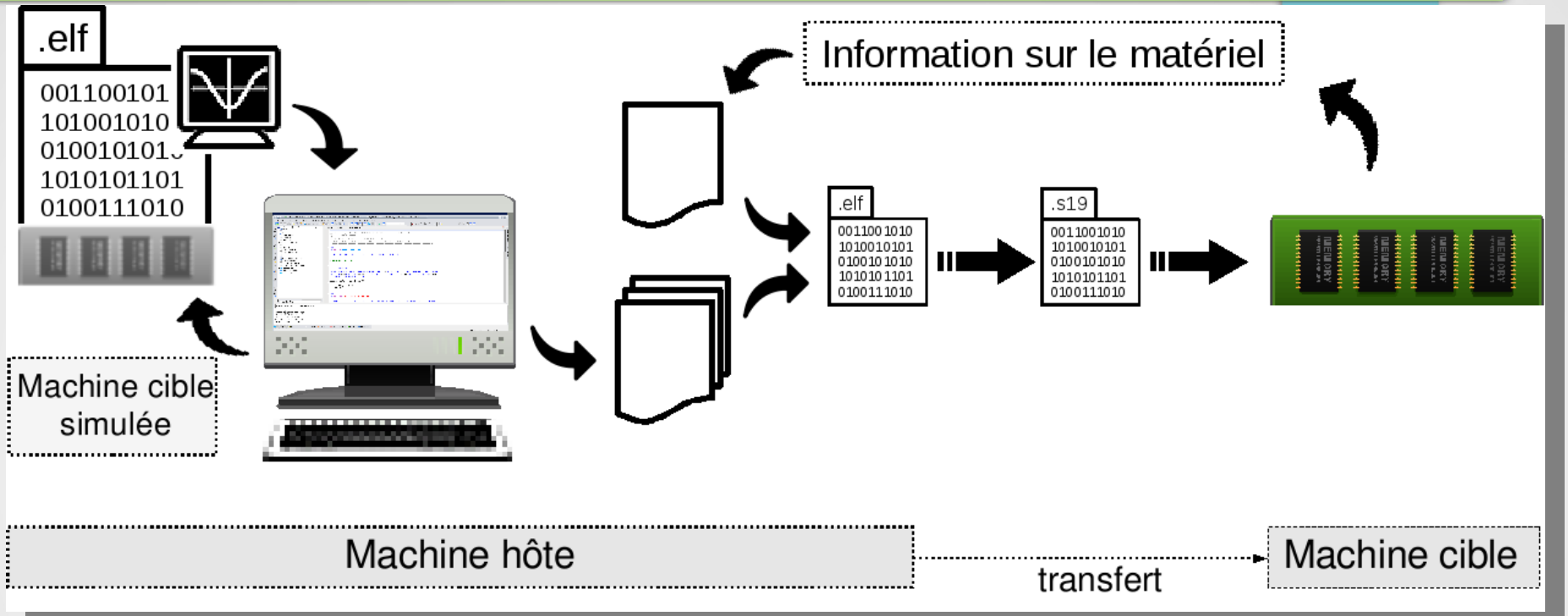


- **Différence forte machine hôte / machine cible**

  Machine cible :

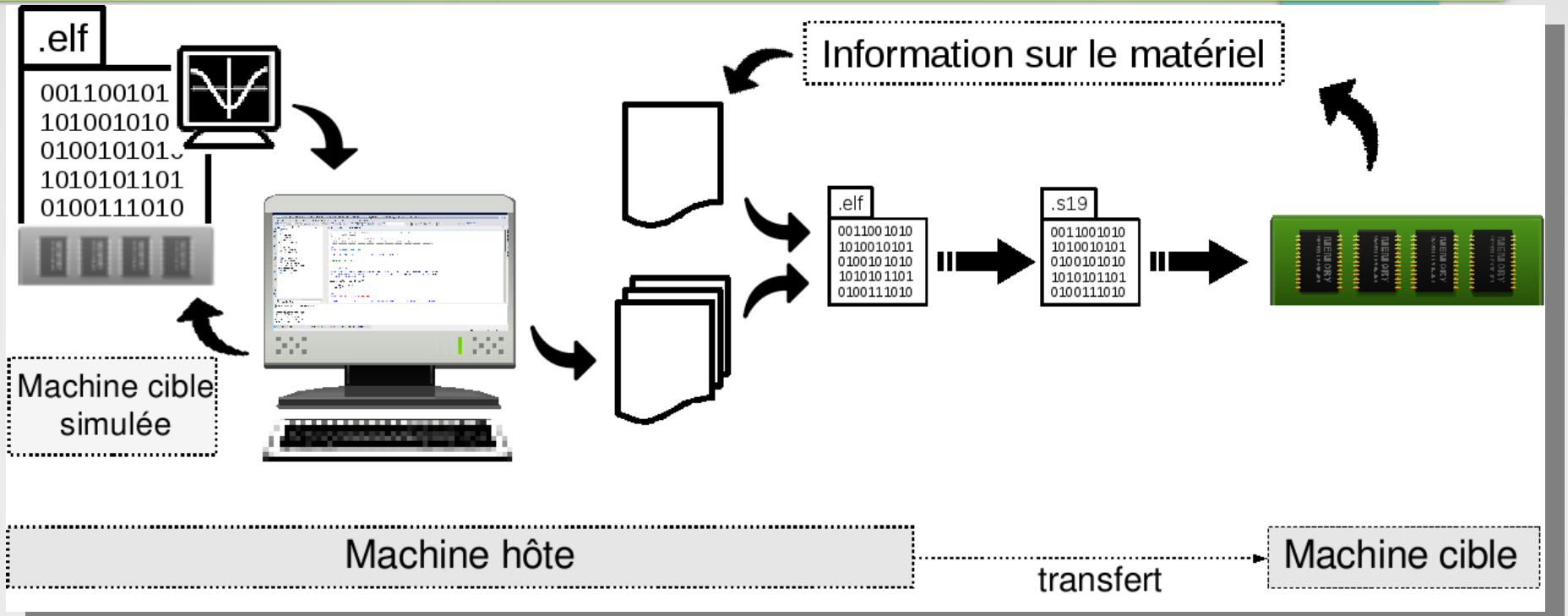
  - Entrées / Sorties ?
  - Difficilement utilisable hors de son **environnement**

Le développement de systèmes temps réel
version simple :-)



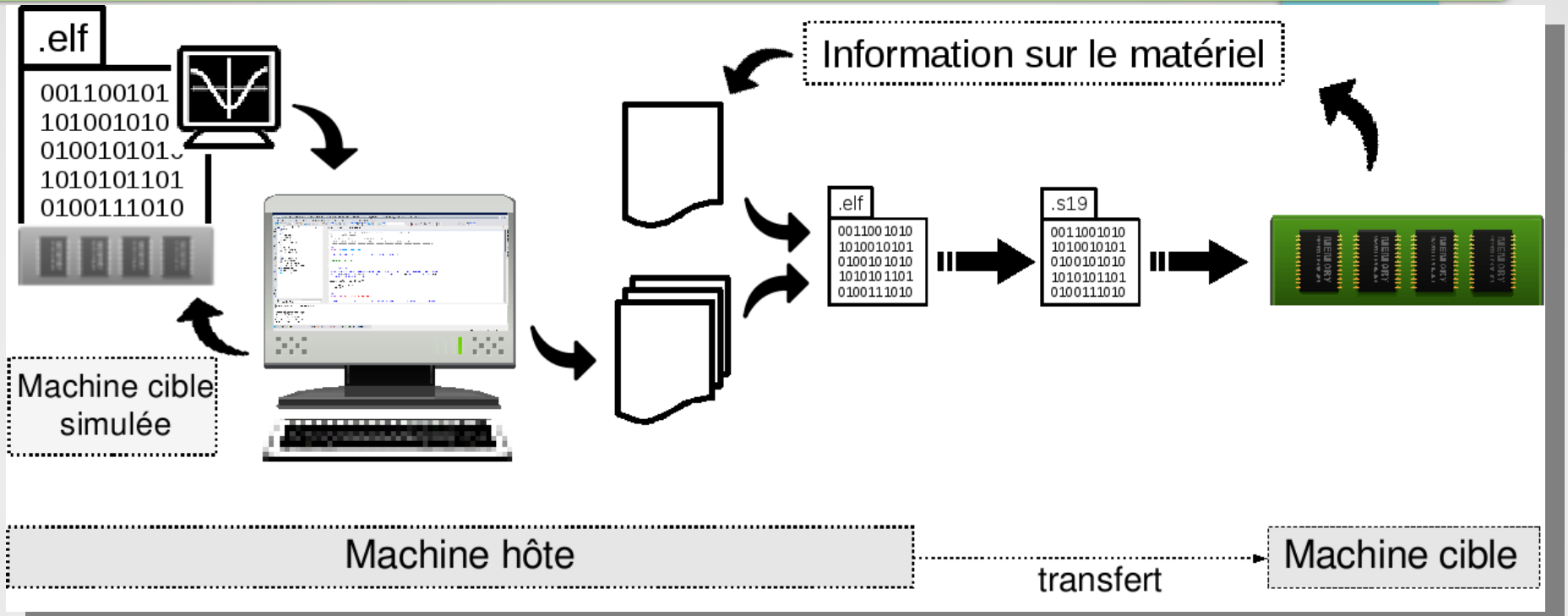
- Méthode de validation (fonction de la criticité)
  - Utilisation de méthodes formelles
  - Tests (critères de couverture)
  - Simulation fonctionnelle (exhaustive ou non) sur machine hôte

Le développement de systèmes temps réel
version simple :-)



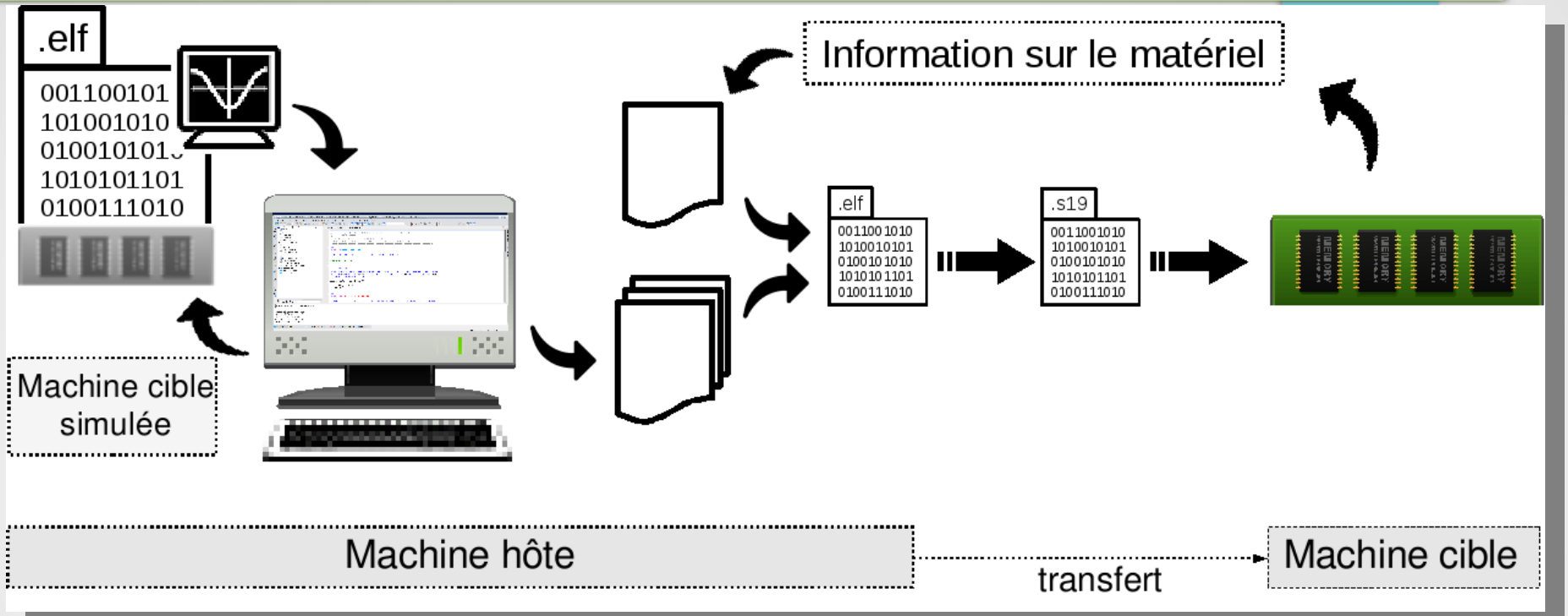
- Critères de validation
  - Temporelle
  - Énergétique
  - Empreinte mémoire
  - ...

Le développement de systèmes temps réel
version simple :-)



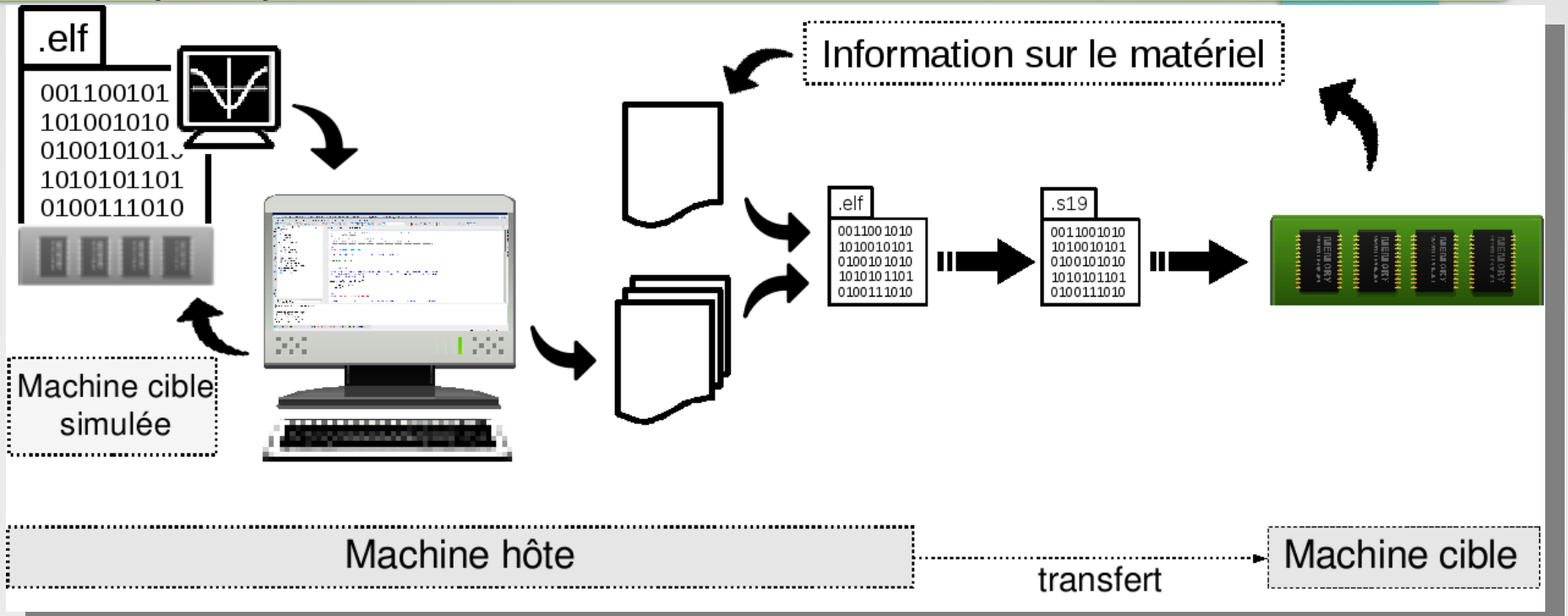
- Cross compilation (compilation croisée)
  - Jeu d'instruction spécifique
  - Configuration matérielle spécifique
    - En particulier mapping mémoire
  - Avec information de débuggage (elf) ou non (s19)

Le développement de systèmes temps réel
version simple :-)



- Cross compilation et validation ?
  - Re-validation du code généré
  - Compilateurs certifiés
    - Répondent à des critères stricts de génération

Le développement de systèmes temps réel
version simple :-)



- Transfert (jtag / SPI / ...)
  - Simple upload
  - Transfert et pilotage
    - Permet le debug
    - Mais l'environnement et le process ?

Contenu du cours

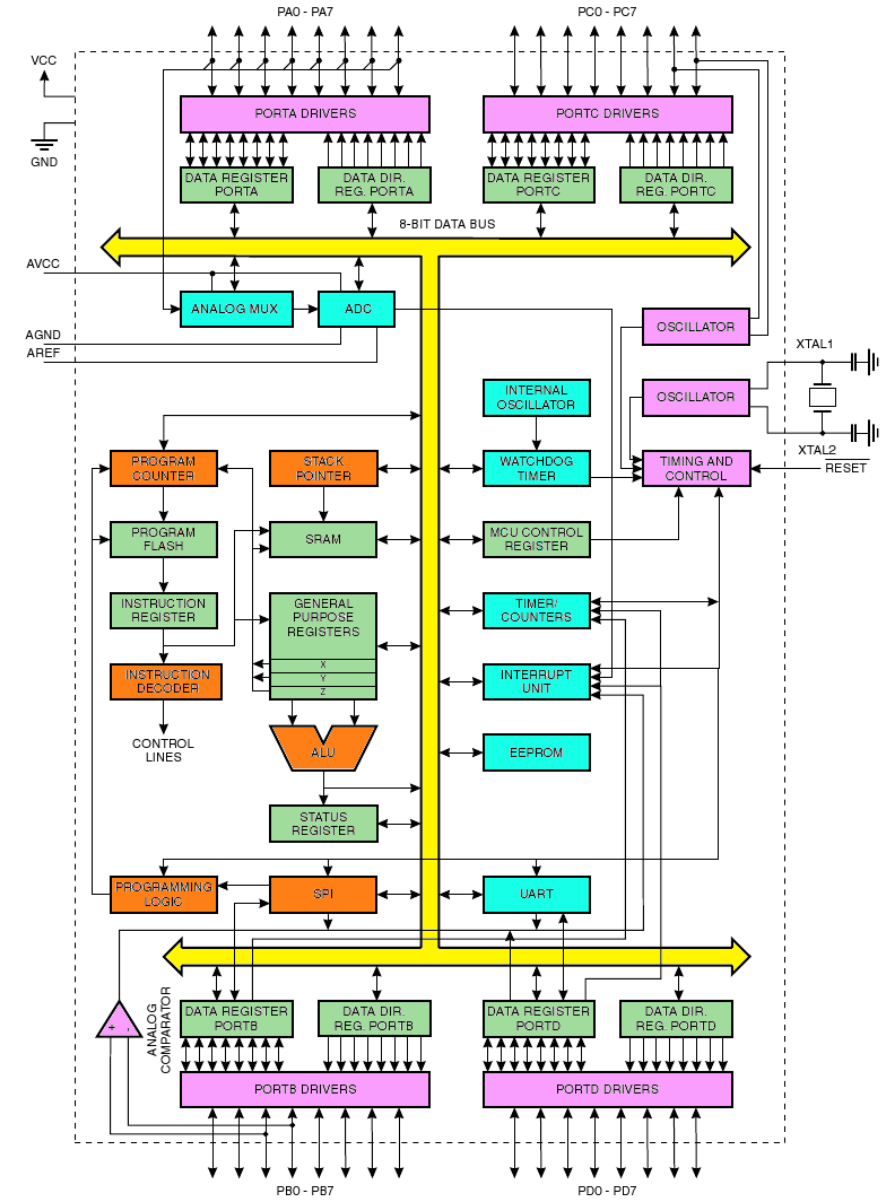
- Généralités
  - Les sytèmes considérés
  - Le développement de systèmes temps réel
- Programmation sans OS
  - **Micro-contrôleur** sans OS, pourquoi ? comment ?
  - Stratégie d'implémentation
    - Programmation sans IT (Synchrone)
    - Programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Micro-contrôleur ?

- Processeur
- Mémoire
- Périphériques
- Bus de communication
- Entrées / Sorties

→ **Dans le même boitier**

Figure 1. The AT90S8535 Block Diagram (From Atmel DataSheet)



Contenu du cours

- Généralités
  - Les sytèmes considérés
  - Le développement de systèmes temps réel
- Programmation sans OS
  - **Micro-contrôleur sans OS, pourquoi ?** comment ?
  - Stratégie d'implémentation
    - Programmation sans IT (Synchrone)
    - Programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Micro-contrôleur sans OS : pourquoi ?

- Un OS prend de la place et du temps de calcul
  - Perte financière (nombre d'exemplaire ?)
- Un OS complique la conception / validation
  - L'OS prend la main ?
  - Combien de temps ?
  - Quand mon code est-il exécuté ?
  - Est-il interrompu ?
- L'OS utilise-t'il toutes les possibilités matérielles ?
  - Ex.: le MSP430 ultra-low-power a 7 modes de mise en veille

Micro-contrôleur sans OS : pourquoi ?

- Un OS n'est pas disponible !
- Vous êtes en train de développer un OS

Micro-contrôleur sans OS : pourquoi ?

- Sans OS
  - Programmation mono-tâche
  - Prédictibilité forte
  - Programmation proche du matériel
    - Optimisation possible
    - Configuration fine et adaptée
  - Gain de place
  - Gain de performance

Contenu du cours

- Généralités
  - Les sytèmes considérés
  - Le développement de systèmes temps réel
- Programmation sans OS
  - **Micro-contrôleur sans OS, pourquoi ? comment ?**
  - Stratégie d'implémentation
    - Programmation sans IT (Synchrone)
    - Programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Micro-contrôleur sans OS : comment ?

- Mettre en place votre environnement de développement
  - Choisir un langage de développement
    - Assembleur
    - C / C++
    - Ada
    - ...
  - Trouver / choisir un compilateur adapté
  - Trouver un *linker* (pour faire le transfert)
  - Trouver / choisir un simulateur si disponible
  - Se procurer les *datasheets* du micro-contrôleur

Micro-contrôleur sans OS : comment ?

- Les datasheets en quelques mots
  - C'est la documentation du micro-contrôleur
  - Sont généralement très conséquentes (567 pages pour le processeur des cartes arduino !)
  - Ne contiennent pas que des choses utiles pour les informaticiens
  - Réverbatif si on ne sait pas ce que l'on cherche

Ce n'est pas un roman donc à moins de vouloir devenir expert des moindres fonctionnalités, ne le lisez pas séquentiellement

Micro-contrôleur sans OS : comment ?

- Informations importantes des datasheets
  - Les ports d'Entrées / Sorties (et Brochages des pattes physiques)
  - La description des périphériques intégrés
    - Timer ? Convertisseurs A/D ? liaison série ?
  - L'organisation de la mémoire
    - Intégrée ou non
  - Les registres

- Informations moins importantes
  - Caractéristiques éléctriques (sauf les consos dans les différents mode de veille)
  - Jeux d'instructions assembleur (a moins que...)

Micro-contrôleur sans OS : comment ?

- Les ports d'Entrées / Sorties
  - Permettent de communiquer avec l'electronique de la machine (notamment les capteurs et les actionneurs)
  - Plus ou moins nombreux
  - Multi fonctionnalités
  - Multi Directionnels

Table 6-1. Input/Output Ports

Port	Input Pins	Output Pins	Bidirectional Pins	Shared Functions
Port A	3	3	2	Timer
Port B	—	8	—	High-order address
Port C	—	—	8	Low-order address and data bus
Port D	—	—	6	Serial communications interface (SCI) and serial peripheral interface (SPI)
Port E	8	—	—	Analog-to-digital (A/D) converter

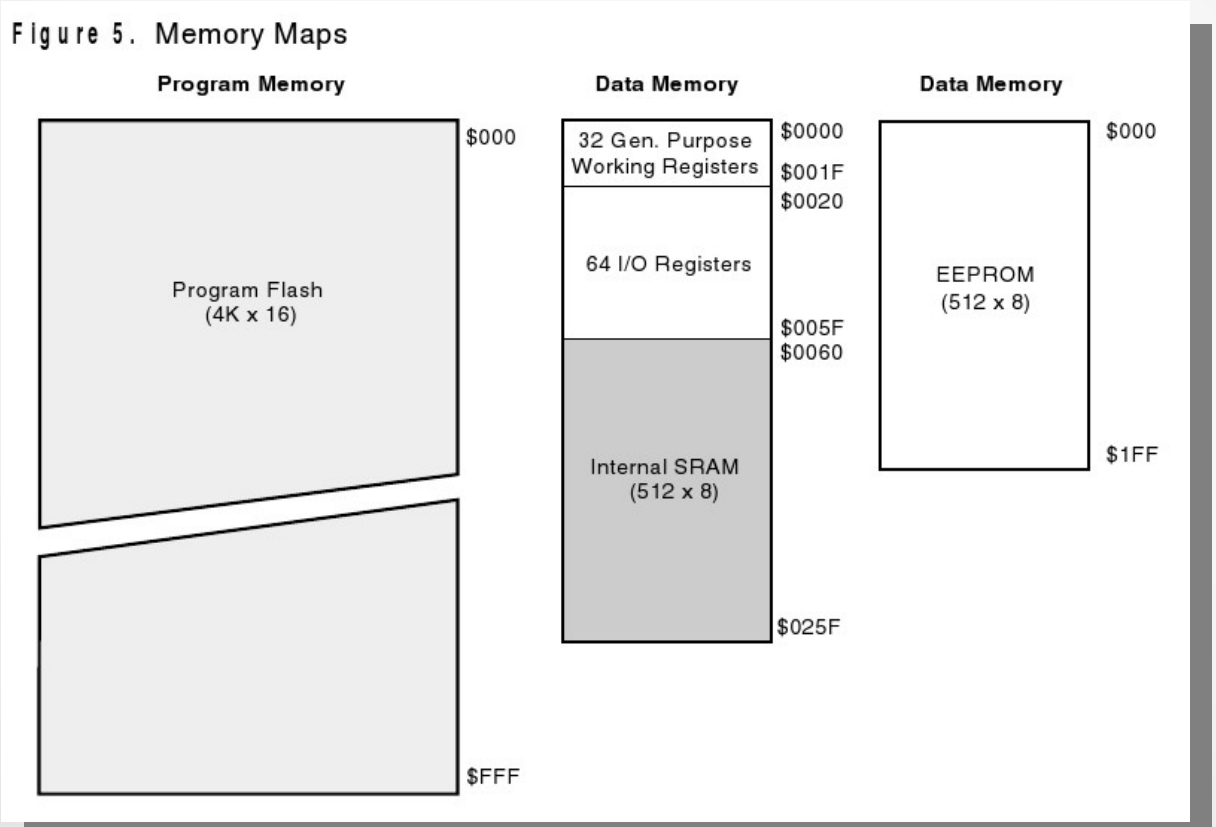
68HC11

Micro-contrôleur sans OS : comment ?

- Les périphériques classiques
  - Timer
    - Permettent de mesurer le temps physique
    - Sont liés à la fréquence physique d'oscillation
    - Programmables et sources d'interruptions ou non
  - Liaison série
    - SPI (Serial Peripheral Interface) – JTAG
    - UART (universal asynchronous receiver transmitter)– RS232
    - ...
    - ➔ Permet souvent le 'transfert' et le debuggage ”*in situ*”
  - Convertisseur Analogique numérique
    - Interface avec des capteurs analogiques

Micro-contrôleur sans OS : la mémoire

- L'organisation de la mémoire ; un exemple



datasheets AT90S8535

The RAM is mapped to $0000 after reset.

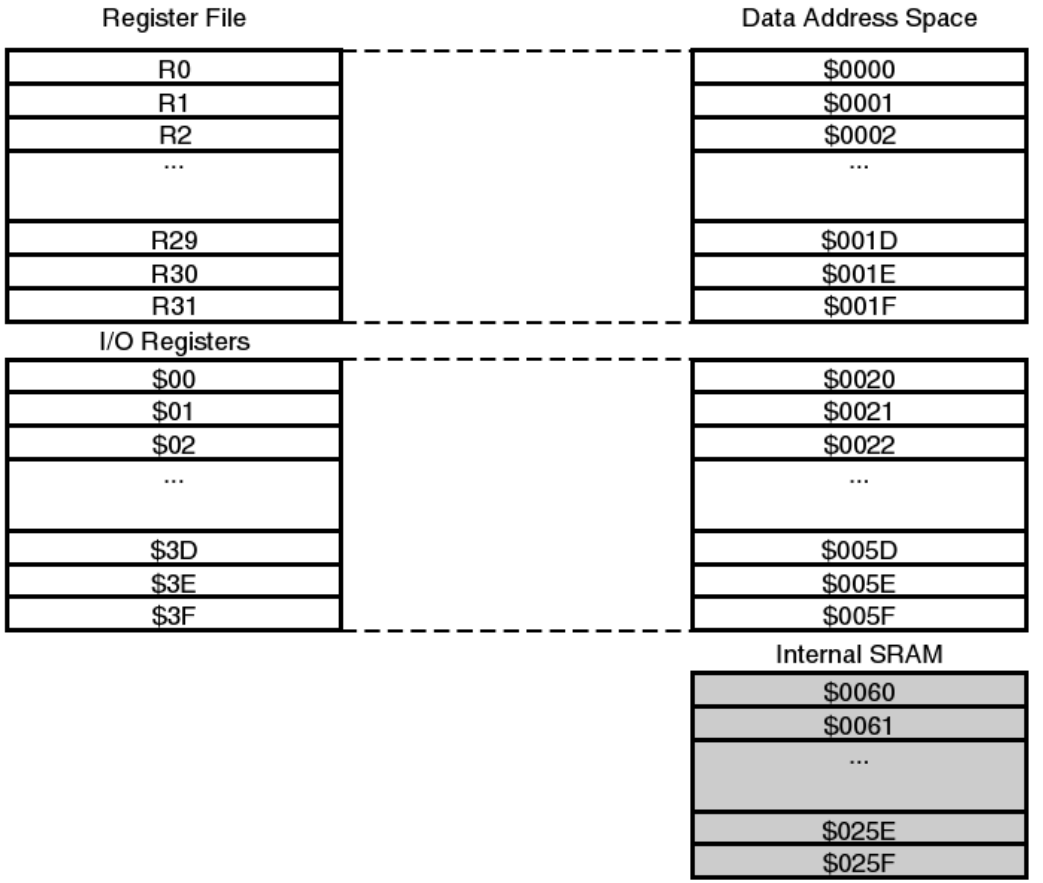
Micro-contrôleur sans OS : la mémoire

- Choisir son mapping
  - Une place pour le programme
    - En RAM pour les premiers tests
    - En EEPROM (ou Flash) pour les tests plus poussés
    - En ROM une fois en production
  - Une place pour les variables
    - En RAM bien sûr
  - Une place pour la pile
    - En RAM mais pas n'importe où (début ou fin ?)
      - Dépend des micro-contrôleurs (implémentation matérielle de push / pop)

Micro-contrôleur sans OS : les registres

- Stockés en RAM
- 2 types
  - Configuration
  - Utilisateurs
- *Peuvent être redondants*

Figure 8. SRAM Organization



Datasheets AT90S8535

Micro-contrôleur sans OS : les registres

- Les registres de configuration
  - Souvent nombreux
  - 8 / 16 / 32 bits
  - À une adresse mémoire particulière
  - À considérer bit par bit

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

bit 4 **INTE**: RB0/INT External Interrupt Enable bit

1 = Enables the RB0/INT external interrupt

0 = Disables the RB0/INT external interrupt

Datasheets PIC16F84A

Micro-contrôleur sans OS : les registres

- Les registres 'utilisateur'
  - 8 / 16 / 32 bits
  - À une adresse mémoire particulière
  - Permettent de stocker les données sur lesquelles travailler (~ variables)
  - Sont utilisés par les instructions de calcul assembleurs

Contenu du cours

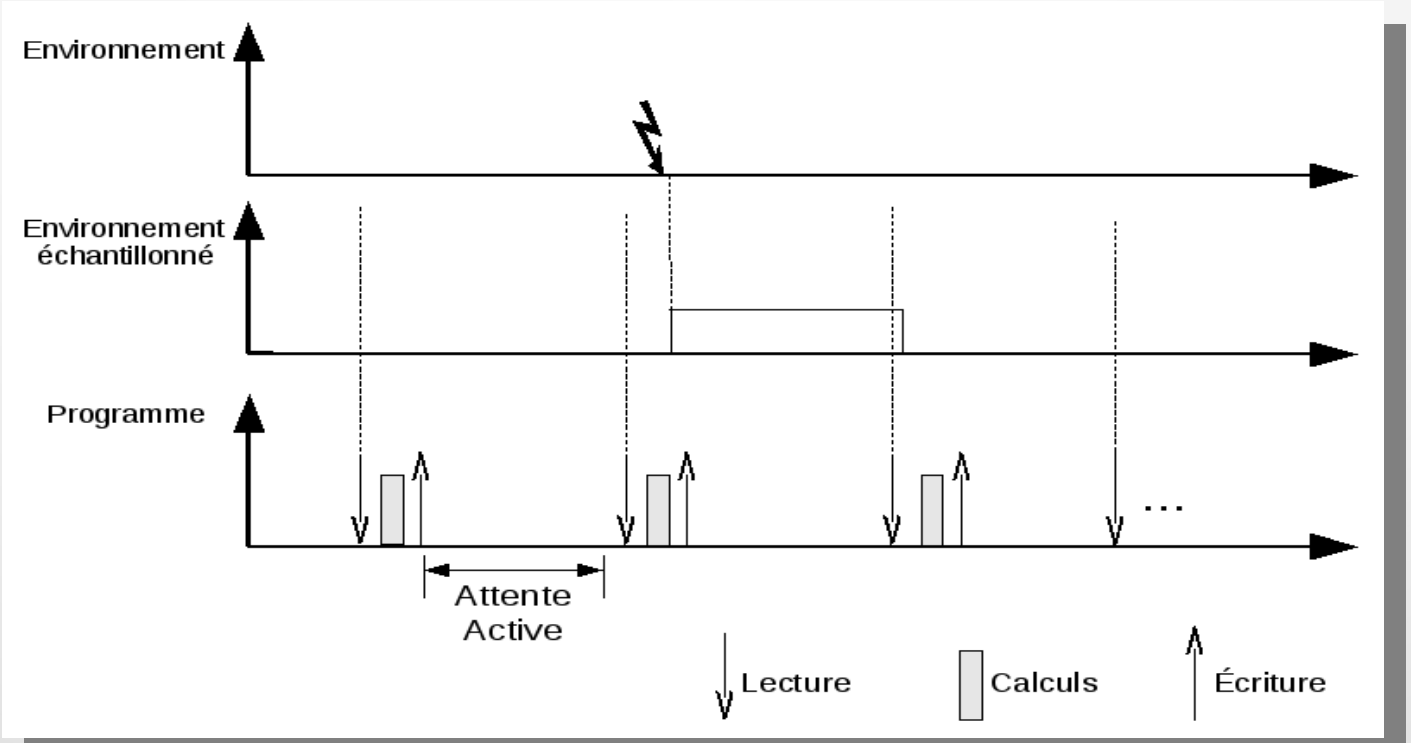
- Généralités
  - Les sytèmes considérés
  - Le développement de systèmes temps réel
- Programmation sans OS
  - Micro-contrôleur sans OS, pourquoi ? comment ?
  - **Stratégie d'implémentation**
    - **Programmation sans IT (Synchrone)**
    - Programmation avec IT (Asynchrone)
- Mise en oeuvre
- Programmation avec un OS temps réel...

Micro-contrôleur sans OS : sans IT

- Stratégie d'implémentation
  1) Sans interruption (Synchrone)
     - Boucle infinie
       - Lecture de tous les capteurs + état système (scrutation)
       - Calculs
       - Écriture actionneurs + état système
       - (attente)

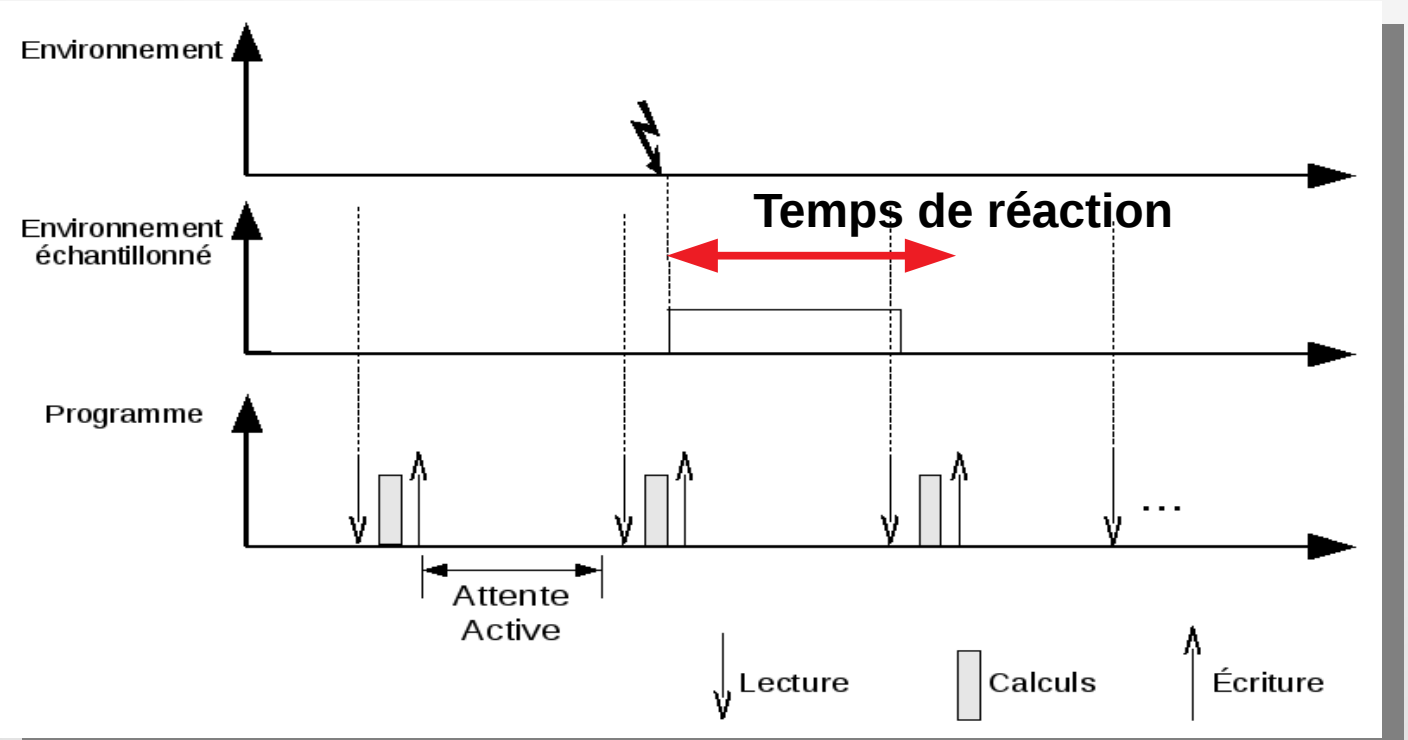
Micro-contrôleur sans OS : sans IT

- Boucle infinie
  - Lecture de tous les capteurs + état système (scrutation)
  - Calculs
  - Écriture actionneurs + état système
  - (attente)



Micro-contrôleur sans OS : sans IT

- Boucle infinie
  - Lecture de tous les capteurs + état système (scrutation)
  - Calculs
  - Écriture actionneurs + état système
  - (attente)



Micro-contrôleur sans OS : sans IT

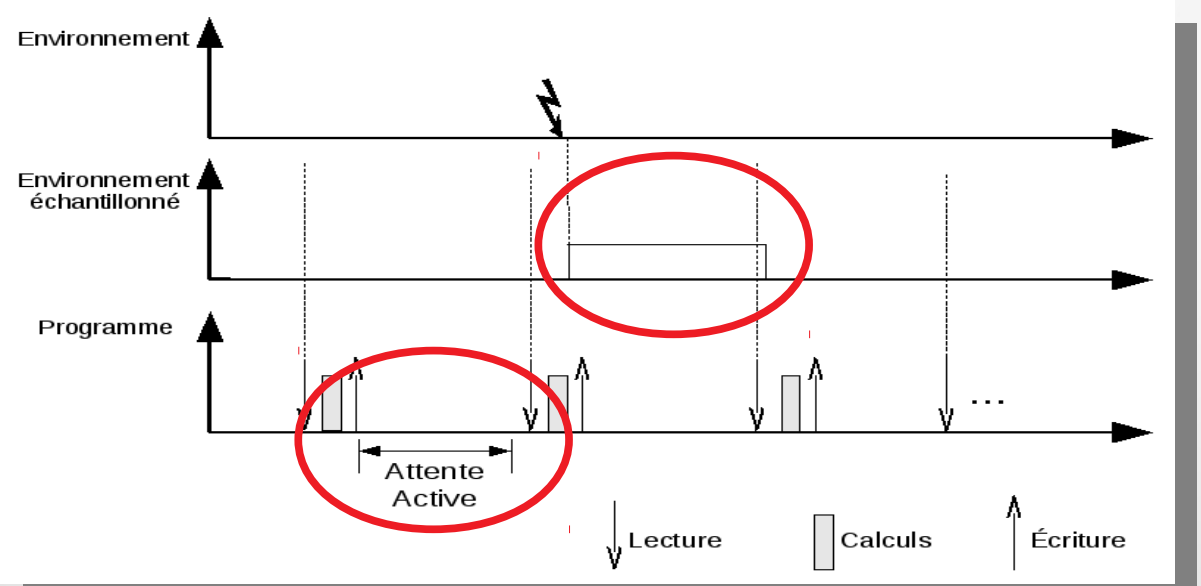
- Stratégie d'implémentation

1) Sans interruption (Synchrone)

- Avantages :
  - Très simple à mettre en oeuvre
  - Calculs des temps de réactions simplifiés
  - Pas / peu de gestion de la pile

Micro-contrôleur sans OS : sans IT

- Stratégie d'implémentation
  1) Sans interruption (Synchrone)
  - Inconvénients :
    - Peu réactif, dépend beaucoup de la durée des calculs
    - Difficile de gérer la consommation électrique (Attente active)

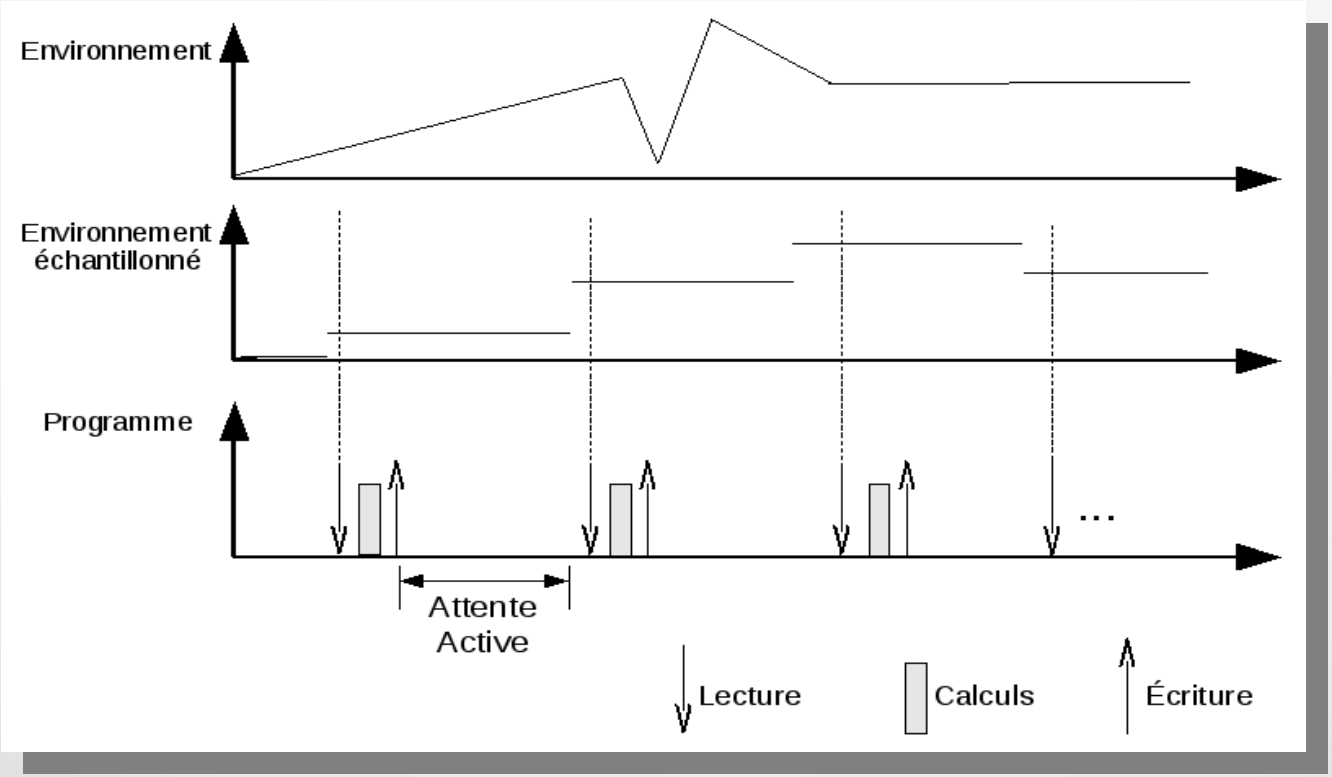


Micro-contrôleur sans OS : sans IT

- Stratégie d'implémentation

1) Sans interruption (Synchrone)

– Inconvénients :
  - Peu réactif, dépend beaucoup de la durée des calculs
  - Difficile de gérer la consommation électrique (Attente active)

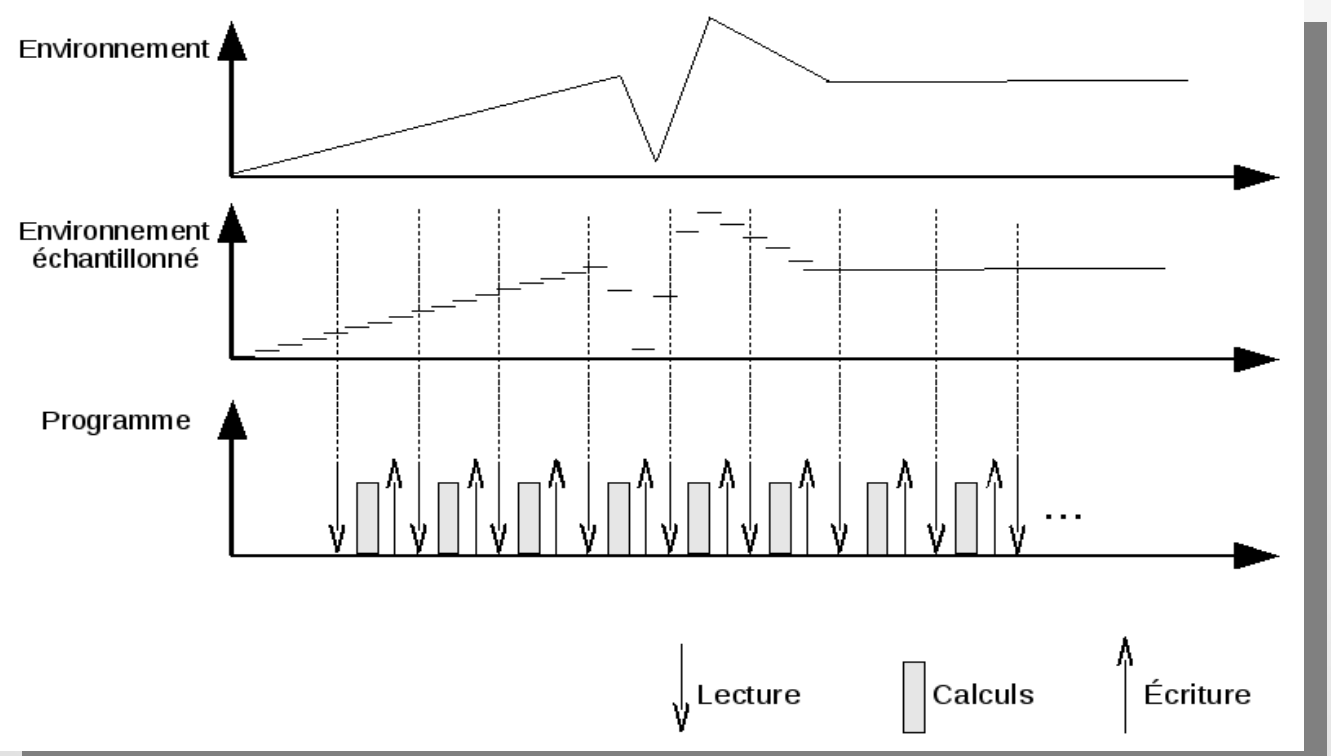


Micro-contrôleur sans OS : sans IT

- Stratégie d'implémentation

1) Sans interruption (Synchrone)

– Inconvénients :
  - Peu réactif, dépend beaucoup de la durée des calculs
  - Difficile de gérer la consommation électrique (Attente active)



Micro-contrôleur sans OS : avec IT

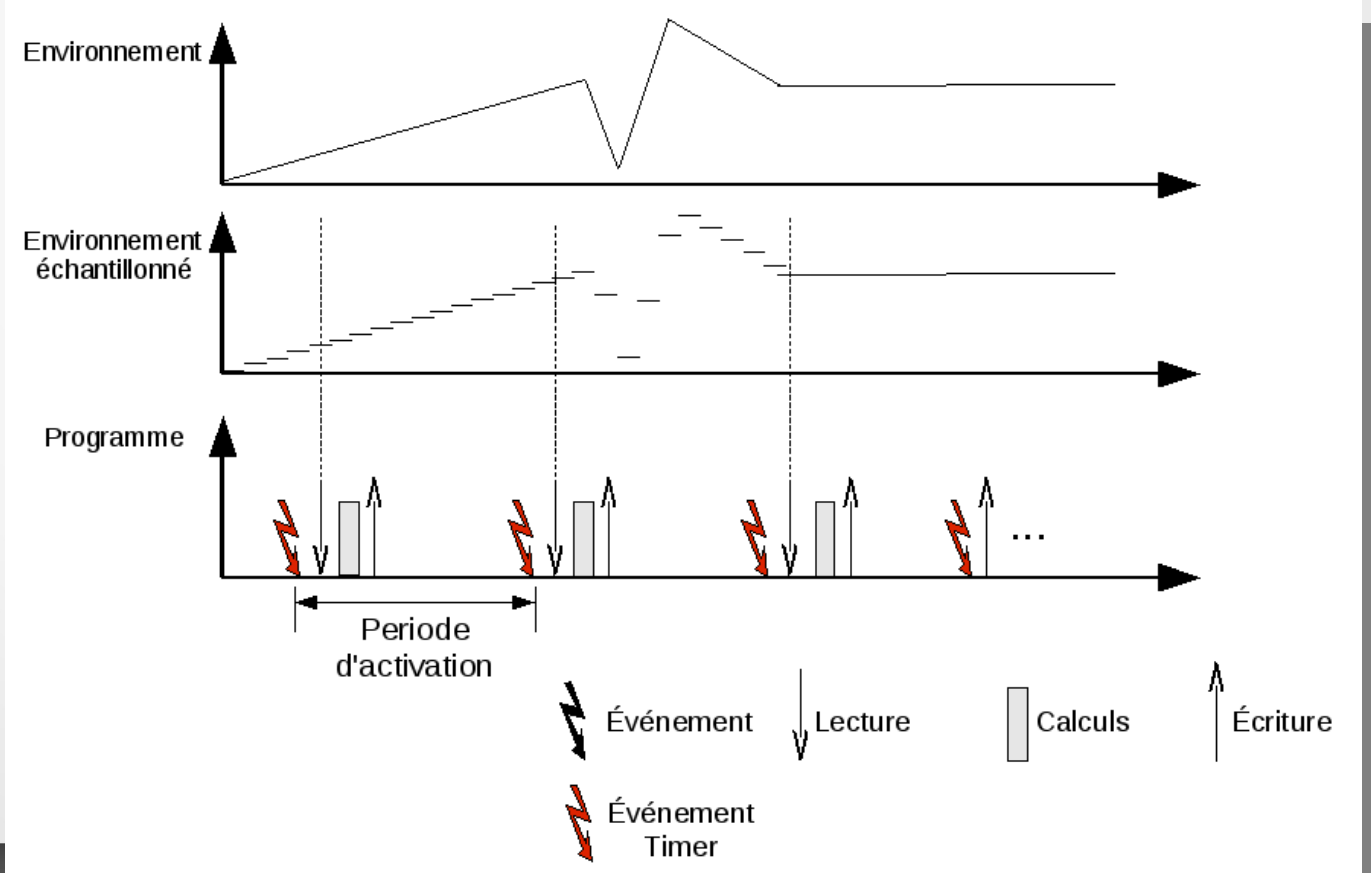
- Choisir sa stratégie d'implémentation

  2) Avec interruption (asynchrone)

  - Idem que sans interruption +
  - Traitement urgence
  - Changement d'états
  - Attente non active

Micro-contrôleur sans OS : avec IT

- Idem que sans interruption +
- Traitement urgence
- Changement d'états
- Attente non active



Micro-contrôleur sans OS : avec IT

- Choisir sa stratégie d'implémentation

2) Avec interruption

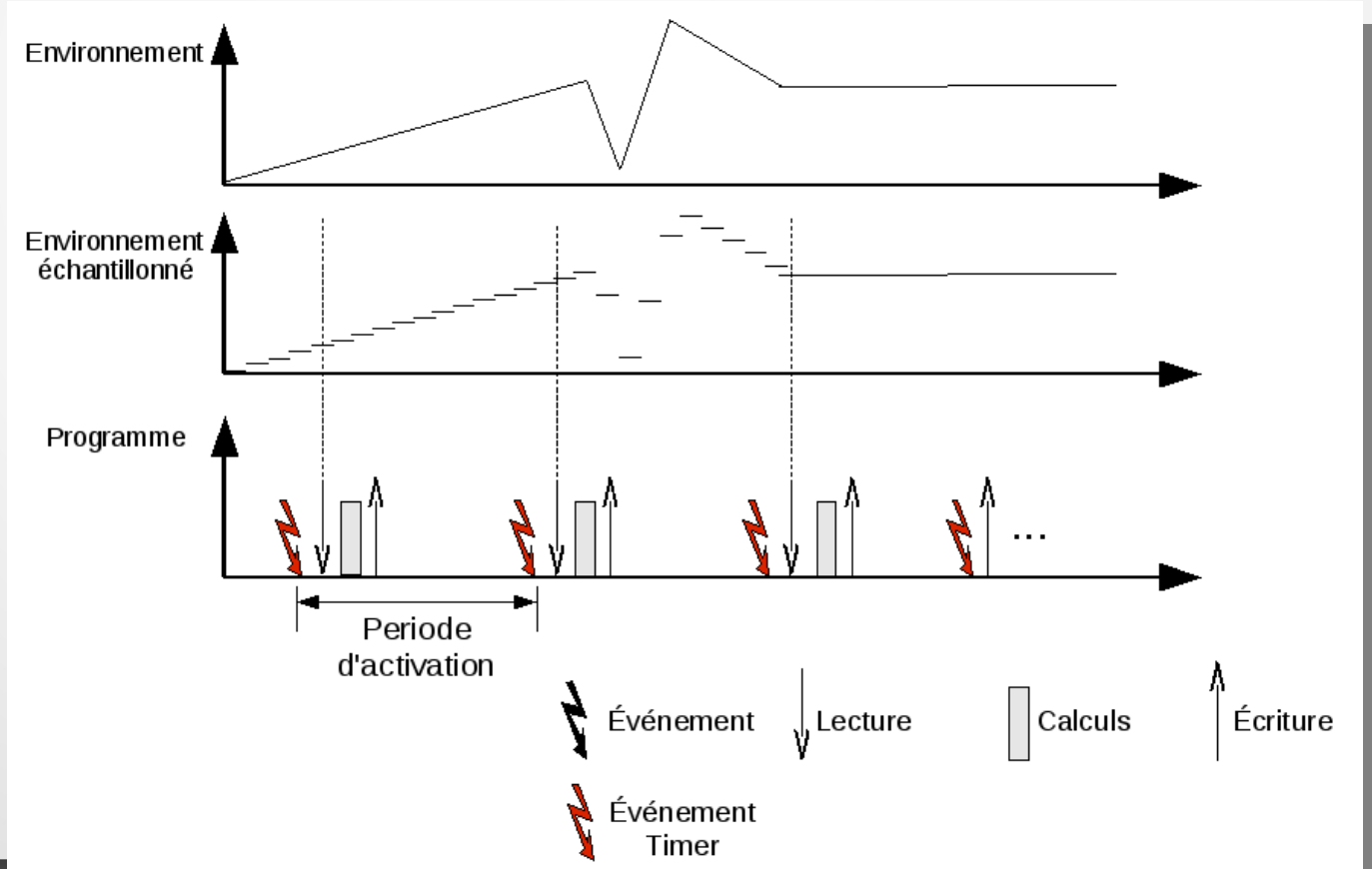
- Idem que sans interruption +
- Traitement urgence
- Changement d'états
- Attente non active

– Avantages :

- Purement réactif
- Permet l'économie d'énergie
- Pas de scrutation non nécessaire

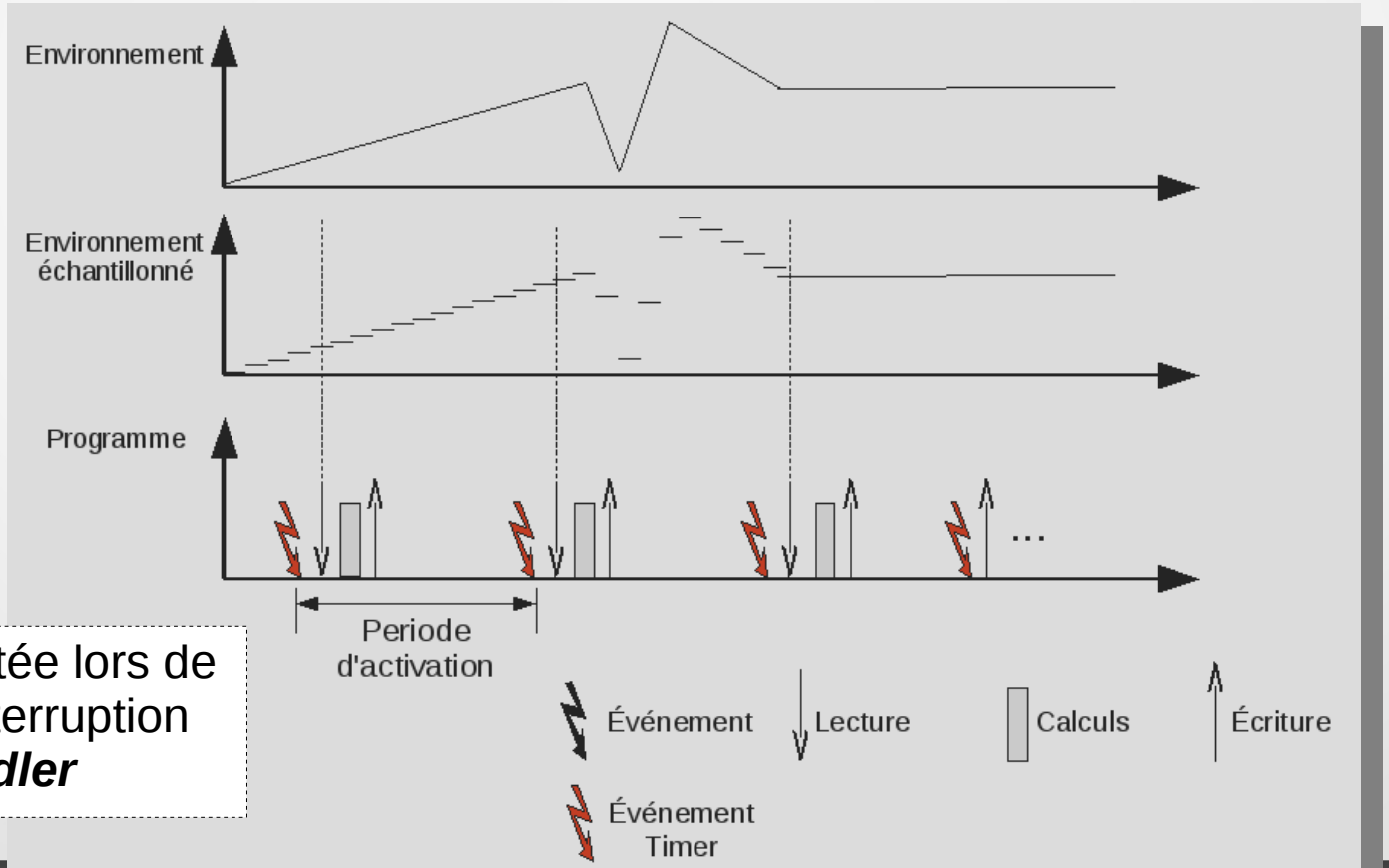
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Avantages :
     - Purement réactif



Micro-contrôleur sans OS : avec IT

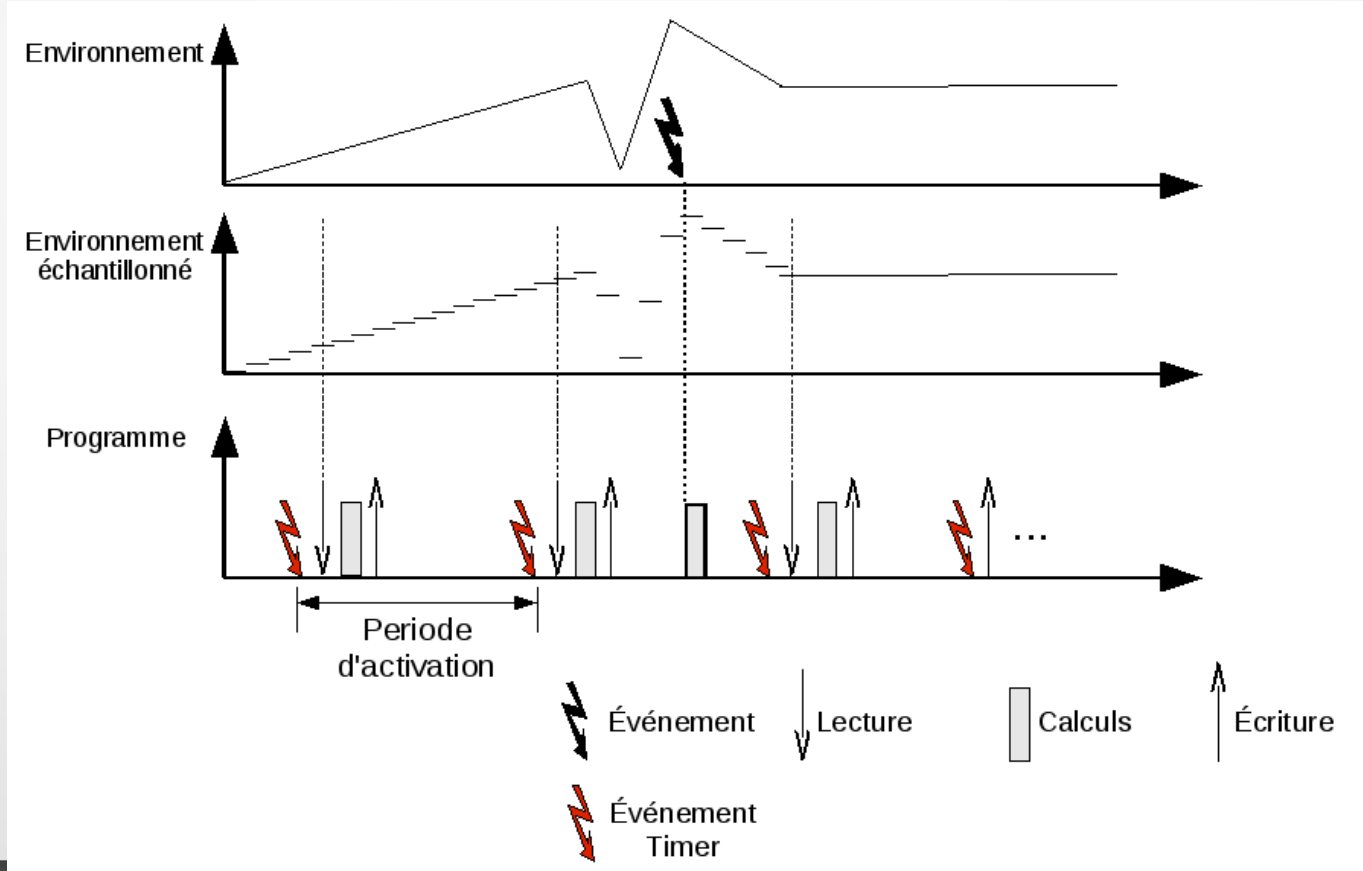
2) Avec interruption
   - Avantages :
     - Purement réactif



La fontion exécutée lors de l'arrivée d'une interruption est appelée ***handler***

Micro-contrôleur sans OS : avec IT

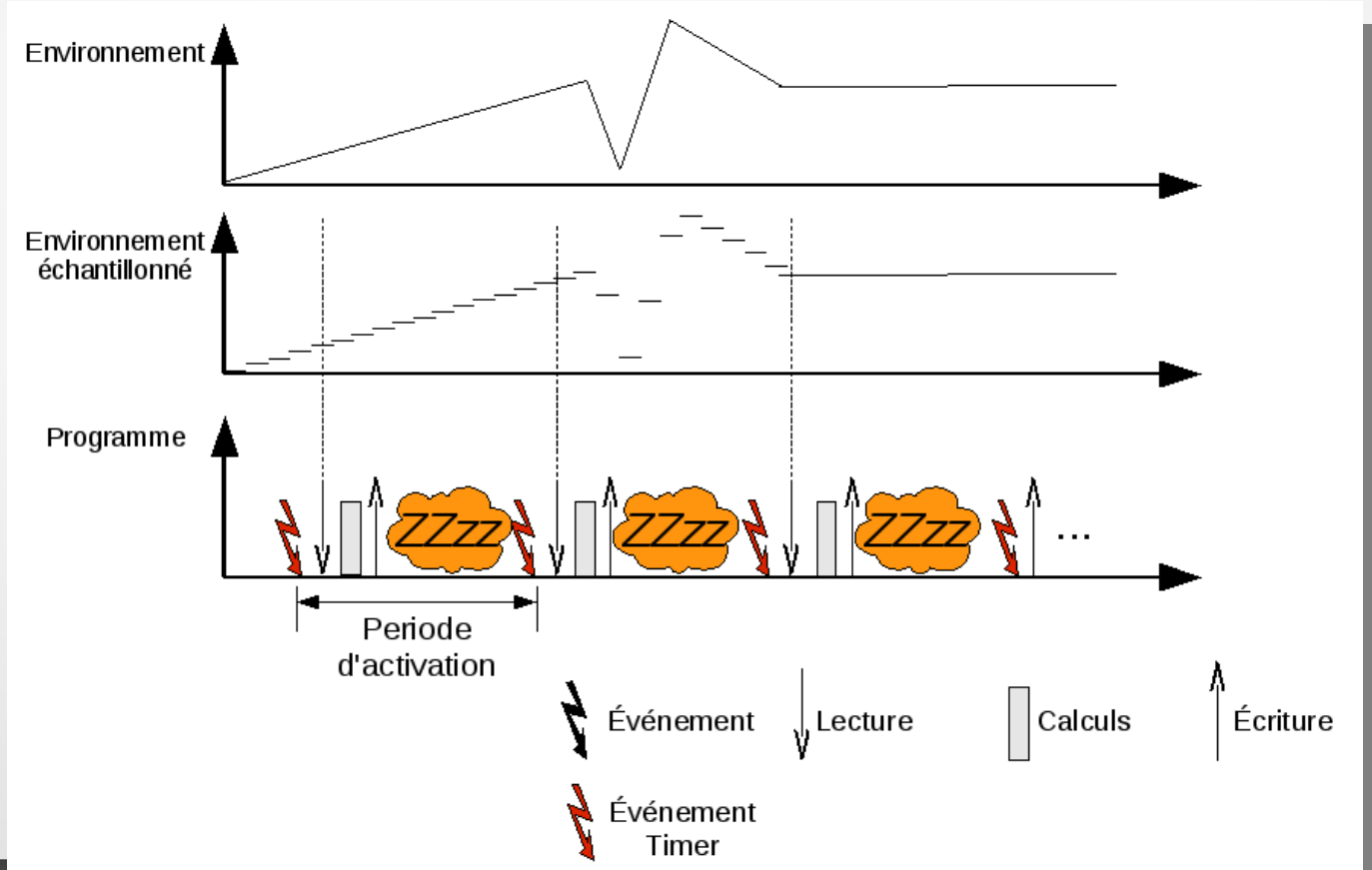
2) Avec interruption
   - Avantages :
     - Purement réactif



Micro-contrôleur sans OS : avec IT

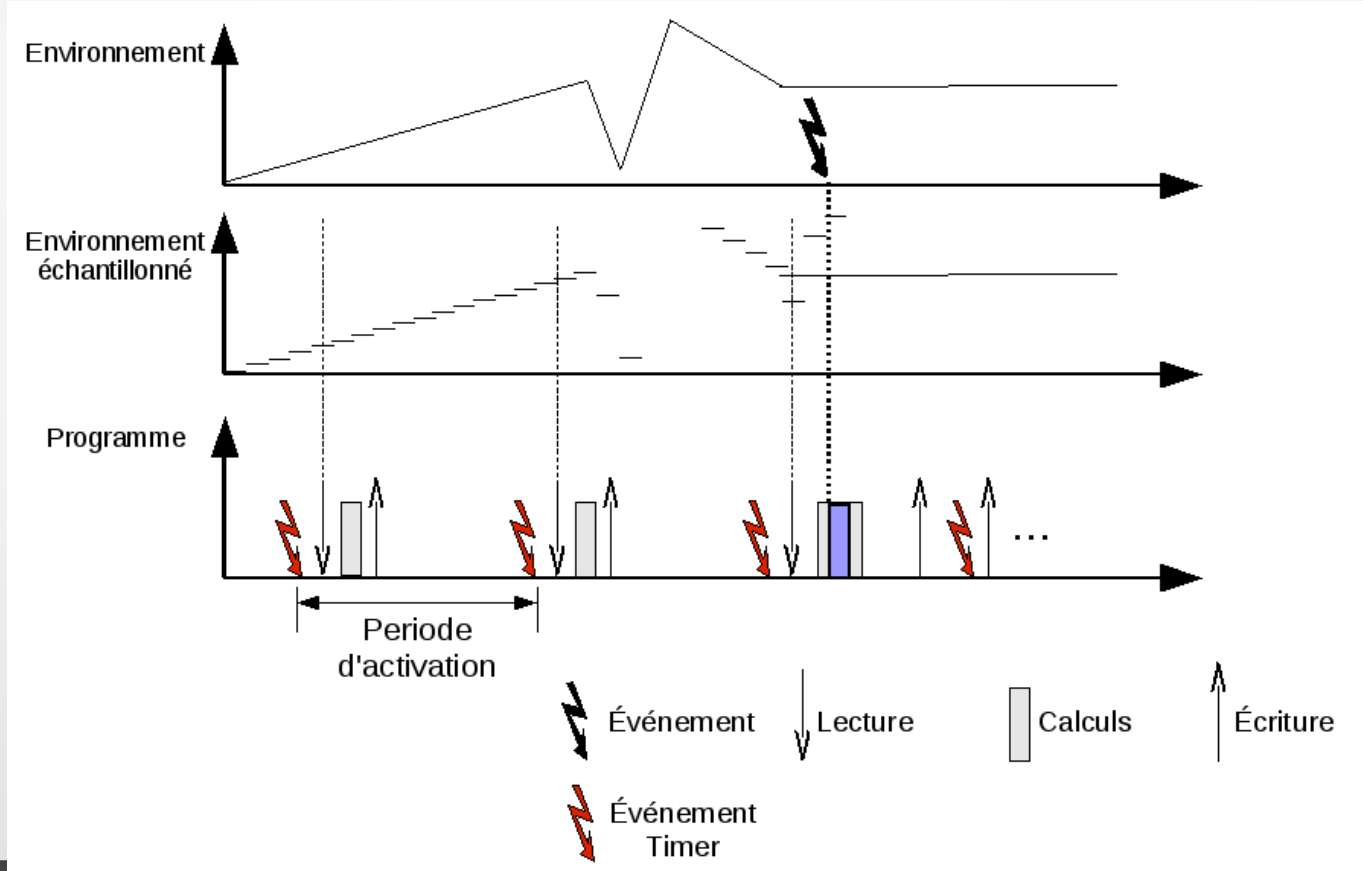
2) Avec interruption

- Avantages :
  - Permet l'économie d'énergie



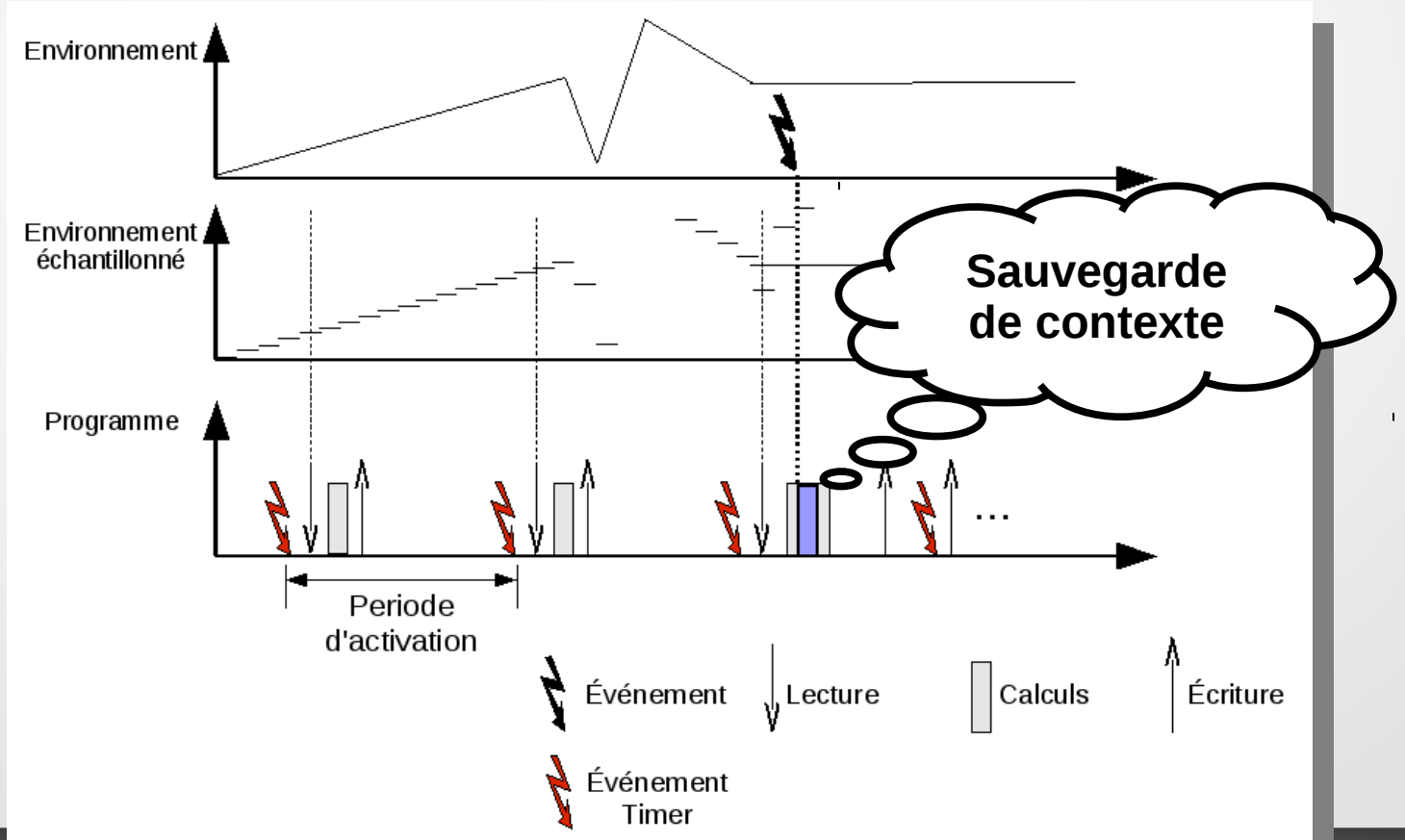
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



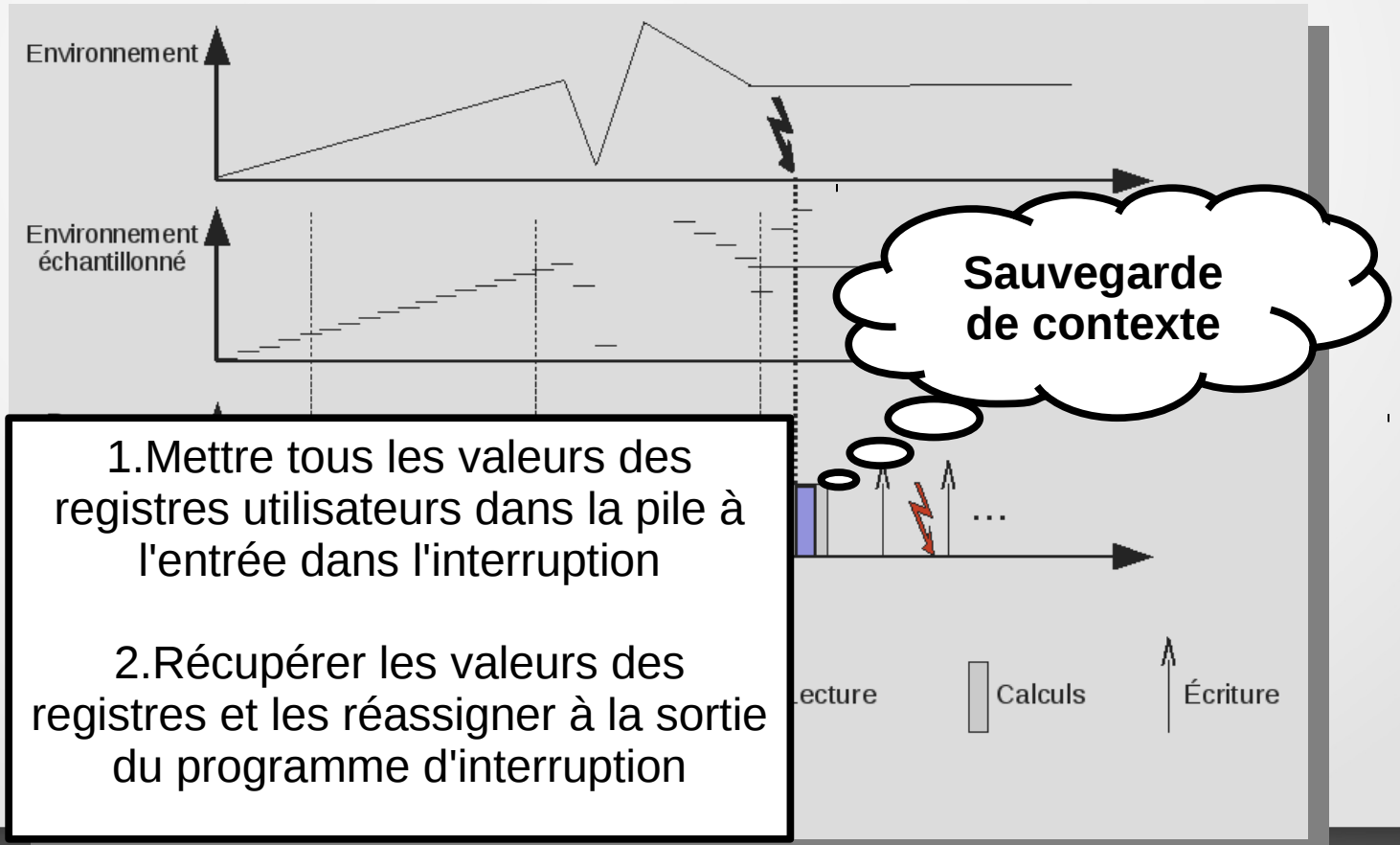
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique

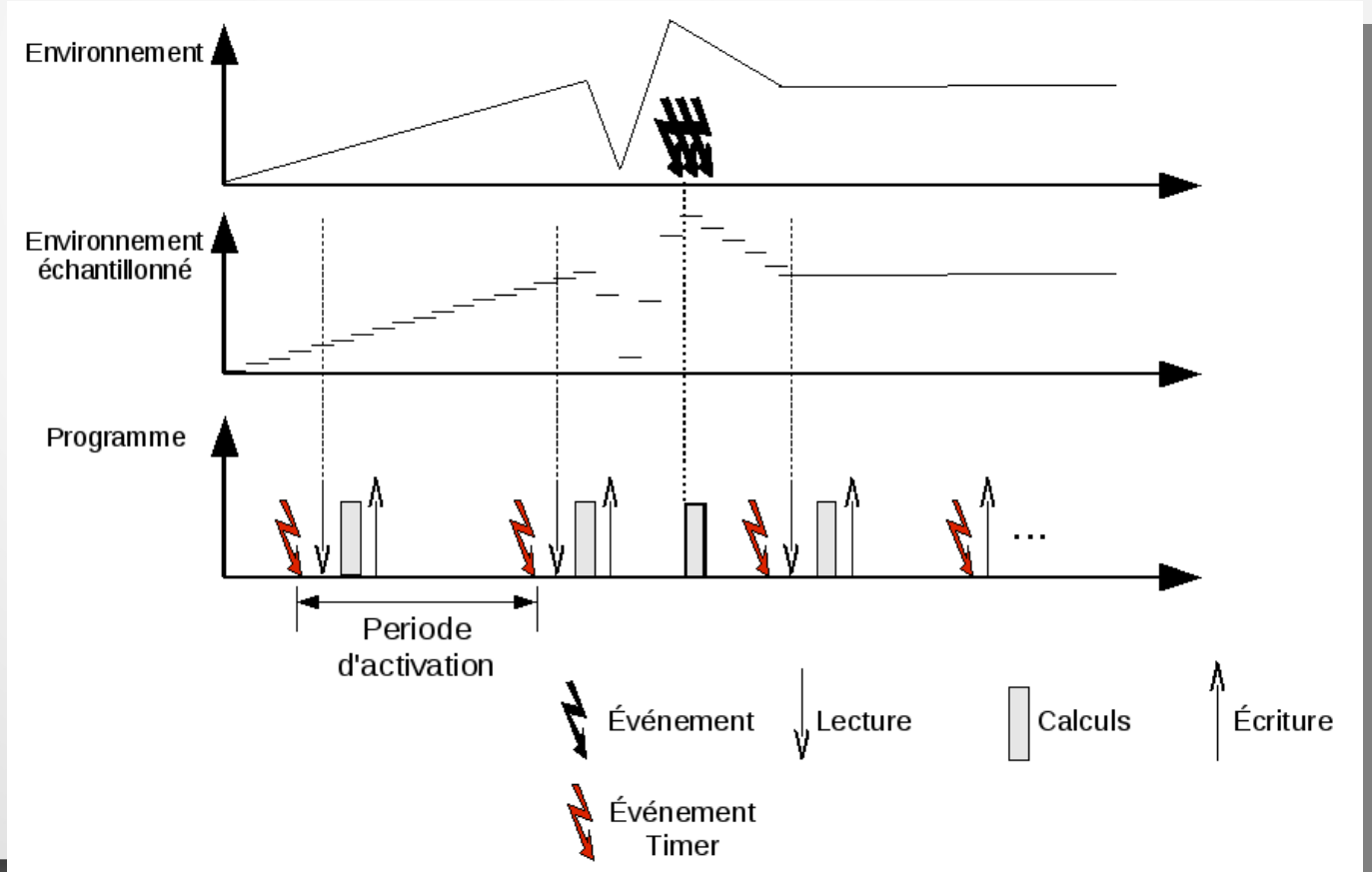


1.Mettre tous les valeurs des registres utilisateurs dans la pile à l'entrée dans l'interruption

2.Récupérer les valeurs des registres et les réassigner à la sortie du programme d'interruption

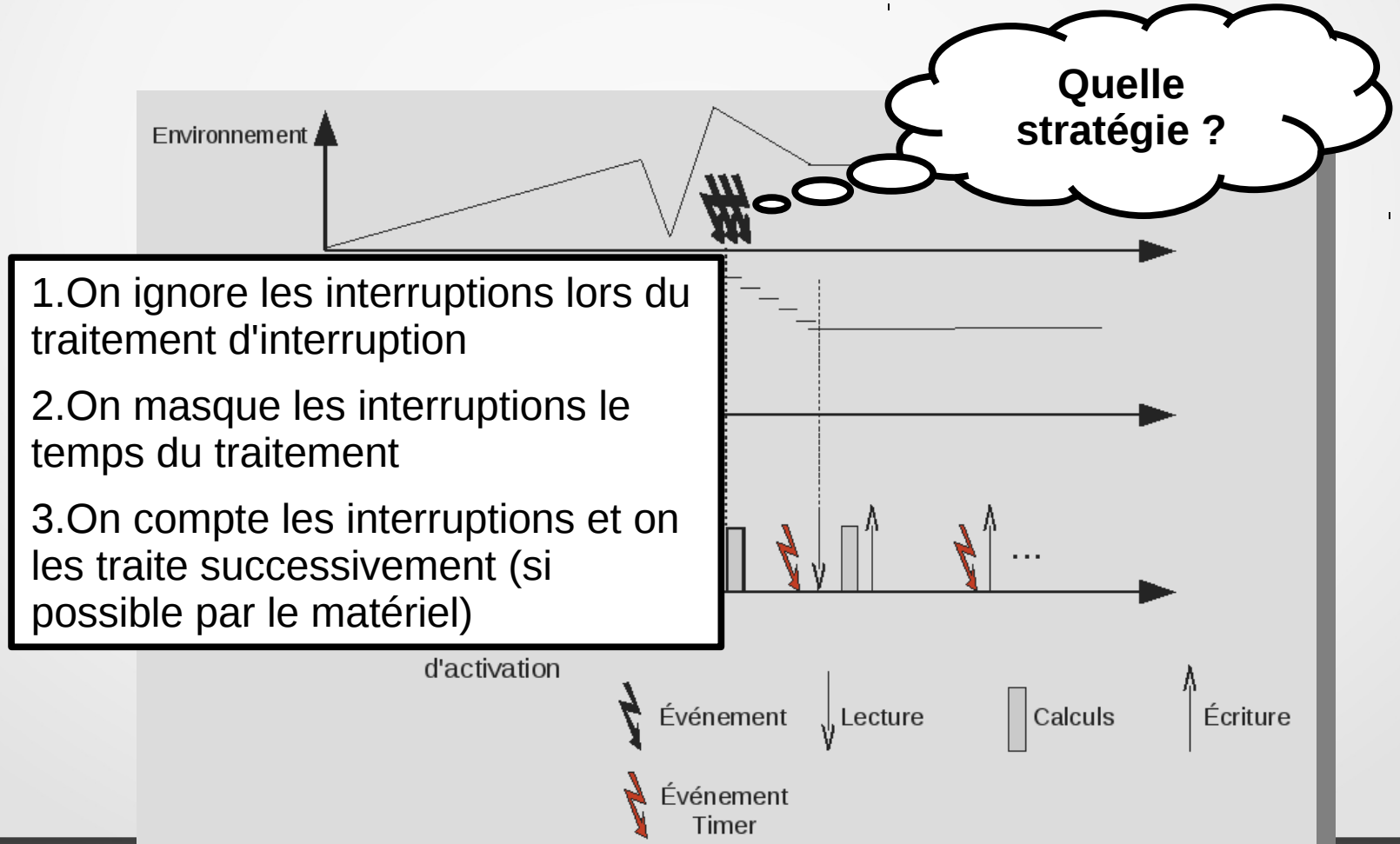
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



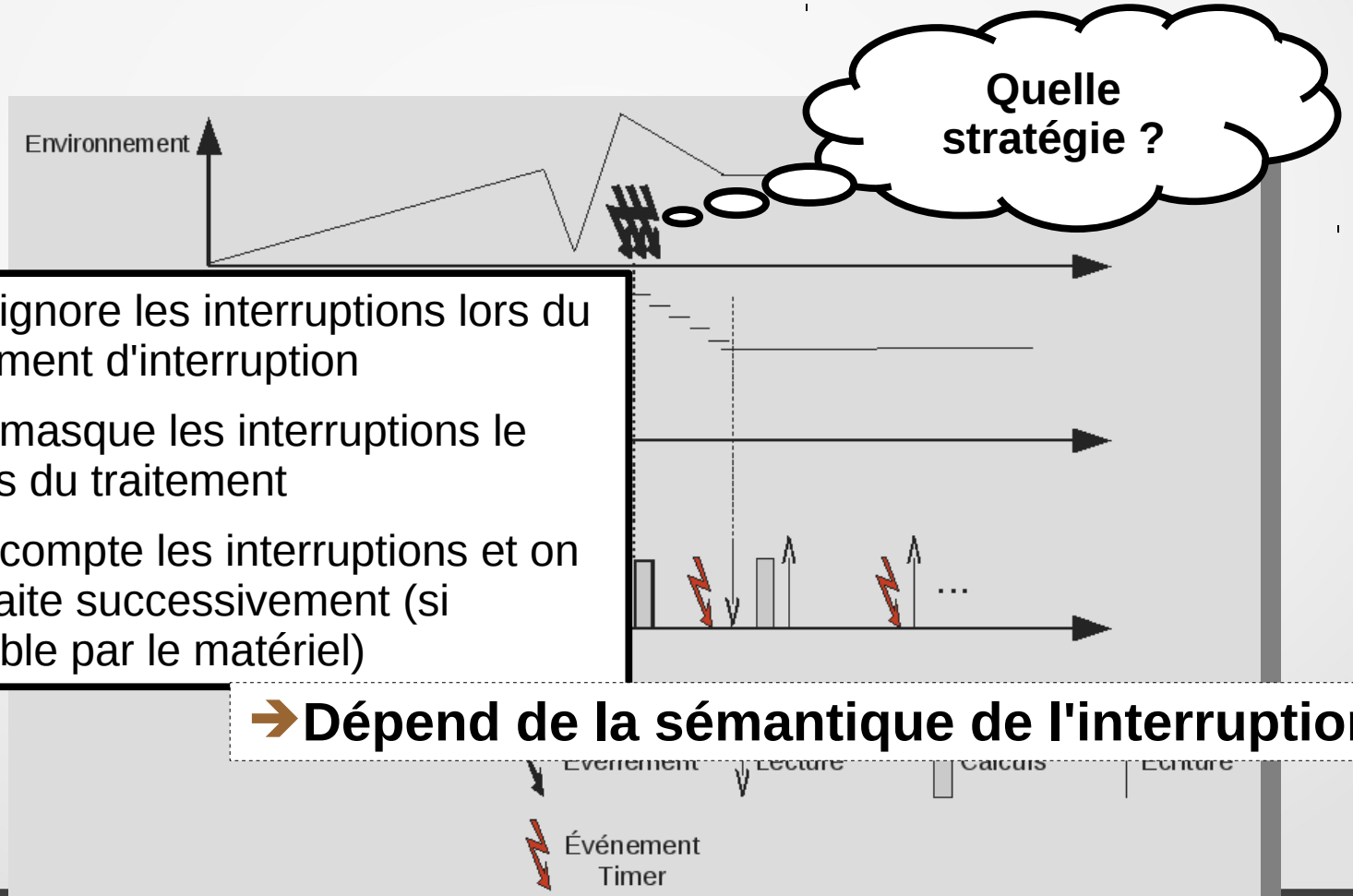
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique

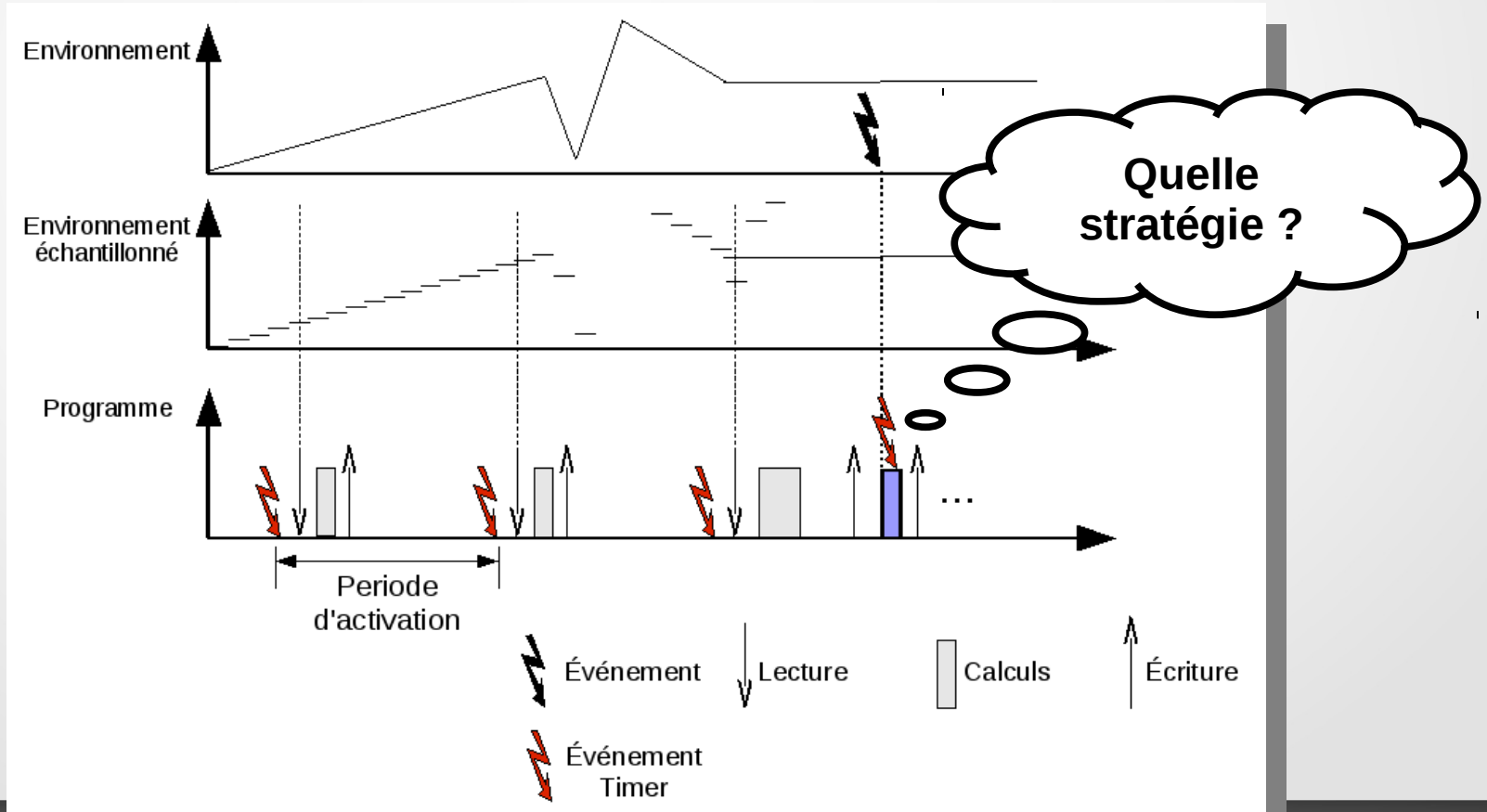


- ✔ On ignore les interruptions lors du traitement d'interruption
- ✔ On masque les interruptions le temps du traitement
- ✔ On compte les interruptions et on les traite successivement (si possible par le matériel)

➔**Dépend de la sémantique de l'interruption**

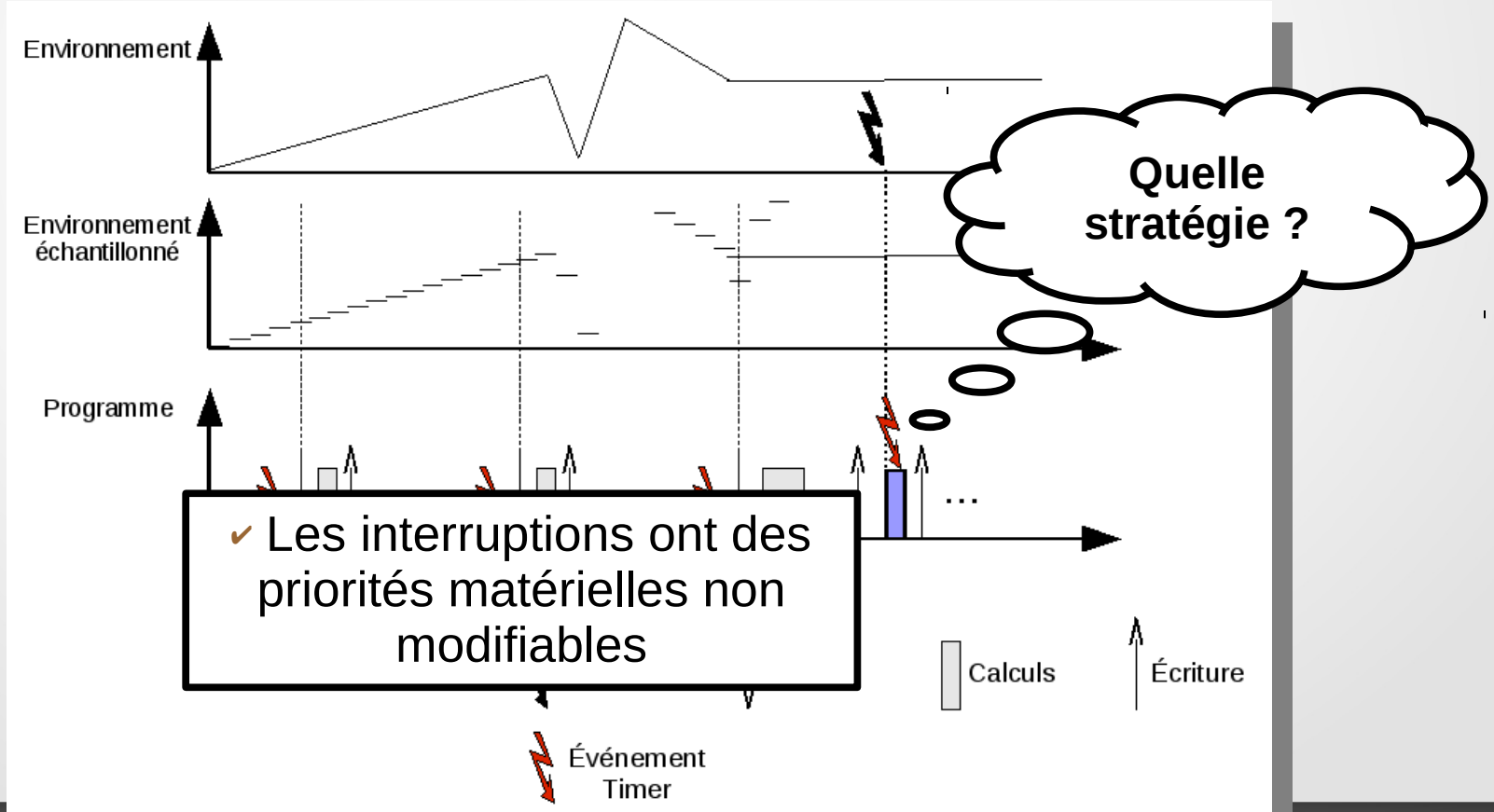
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



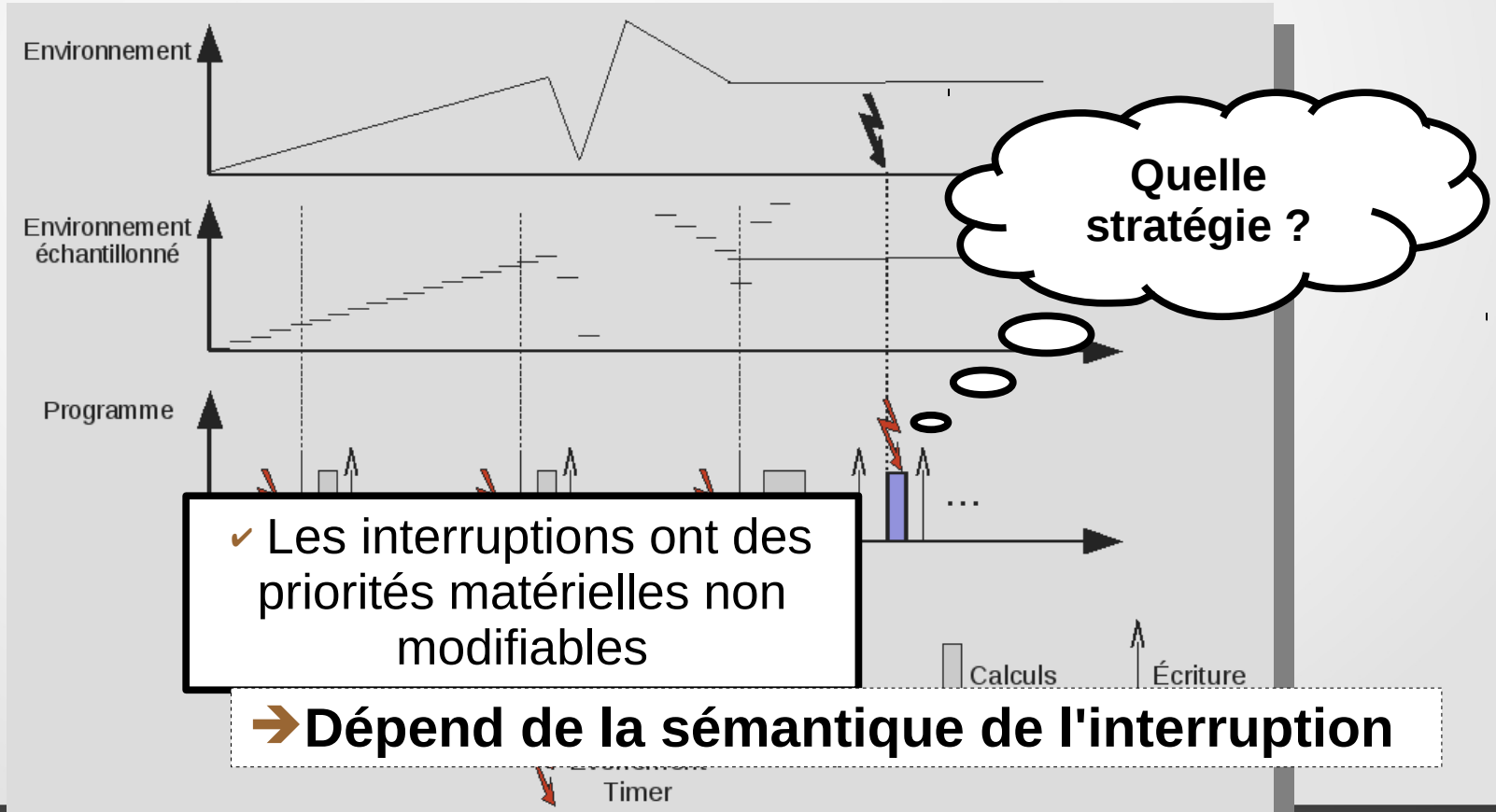
Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique



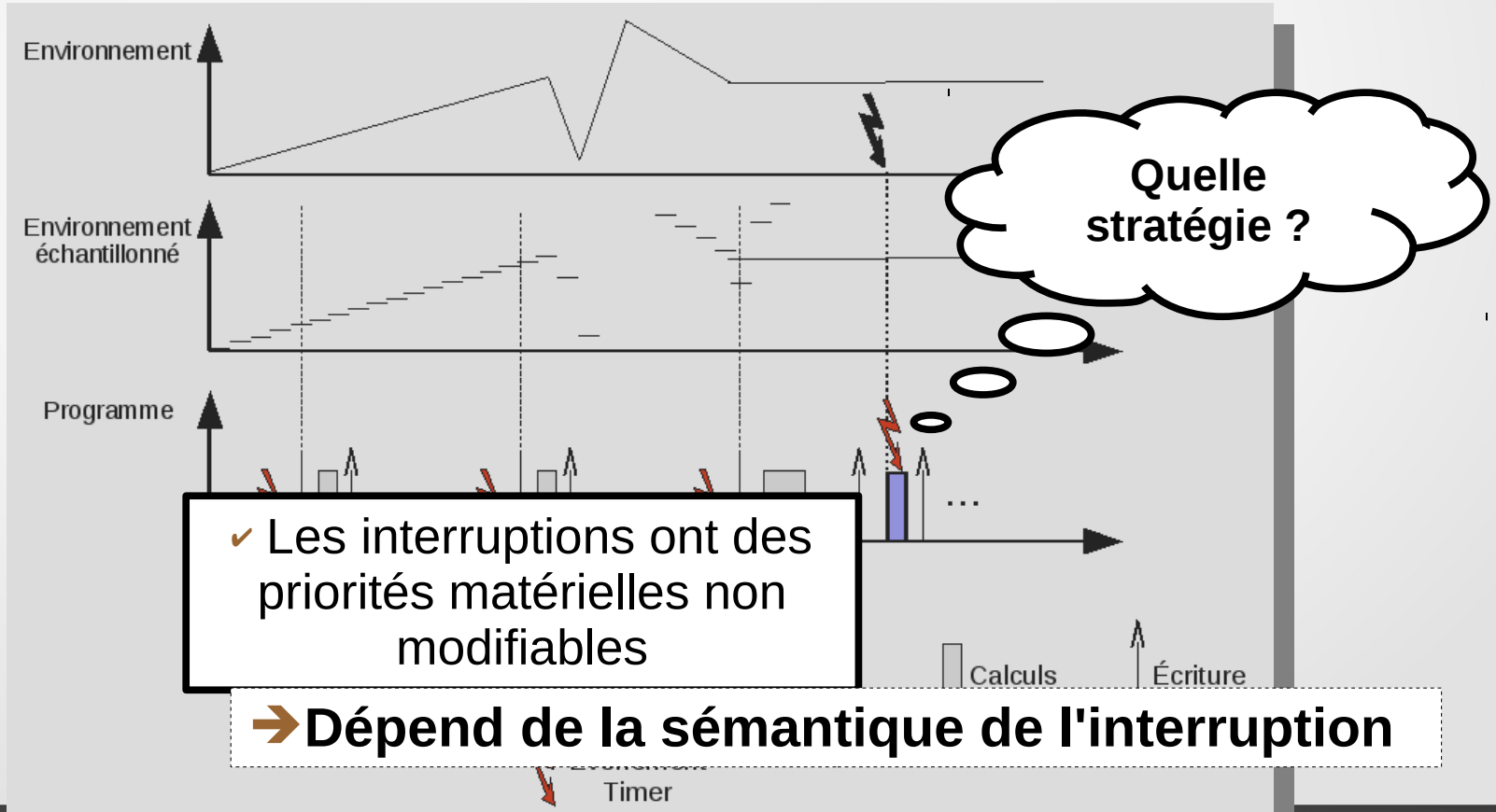
✔ Les interruptions ont des priorités matérielles non modifiables

➔**Dépend de la sémantique de l'interruption**

Micro-contrôleur sans OS : avec IT

2) Avec interruption
   - Inconvénient :
     - Demande un peu de technique

Limite du sans OS ?



✔ Les interruptions ont des priorités matérielles non modifiables

➔**Dépend de la sémantique de l'interruption**

Contenu du cours

- Généralités
  - Les sytèmes considérés
  - Le développement de systèmes temps réel
- Programmation sans OS
  - Micro-contrôleur sans OS, pourquoi ? comment ?
  - Stratégie d'implémentation
    - Programmation sans IT (Synchrone)
    - Programmation avec IT (Asynchrone)
- **Mise en oeuvre**
- Programmation avec un OS temps réel...

Mise en oeuvre: Le micro-contrôleur

- Atmel Atmega328P (16MHz)
  - Architecture RISC
  - 32Ko de Flash / 1Ko d'EEPROM, 2Ko de RAM
  - 23 entrées / Sorties programmables
  - Communication UART et SPI
  - 2 timers 8 bits et 1 timer 16 bits
  - 6 modes de veilles
  - ... (voir datasheet)

Mise en oeuvre: La chaîne de compilation

- AVR-GCC and co
  - Cross compilateur basé sur gcc
  - Linker (avrudude) , librairies pour architecture spécifiques (avr-libc)
  - Windows / Linux
  - ➔ http://www.avrfreaks.net/wiki/index.php/Documentation:AVR_GCC
- ICCAVR
  - Cross compilateur propriétaire Atmel
  - Windows
  - ➔ CD fourni

Mise en oeuvre: La chaîne de compilation

- Définition des handlers différentes selon le compilateurs / linkers
  - AVR-gcc

```
//handler de l'interruption nommées INT_COMPA
ISR(INT_COMPA_vect)
{
   // traiter l'interruption
}
```

  - ICCAVR

```
 //handler de l'interruption numéro NUM
#pragma nom_fonction_handler ISR:NUM
[...]
 nom_fonction_handler
 {
    // traiter l'interruption
 }
```

Mise en oeuvre: Simuler

- SimulAVR
  - Simul le jeux d'instruction RISC AVR
  - Permet de visualiser l'état de la mémoire et des registres en pas à pas...ou pas
  - Windows / Linux
  - ➔ http://savannah.nongnu.org/projects/simulavr/
  - ➔ http://www.nongnu.org/avr-libc/user-manual/install_tools.html
- AVRStudio
  - Idem SimulAVR MAIS Solution professionnelle
  - Windows
  - ➔ CD fourni

Mise en oeuvre: *Linker* et transférer

- Avr-gcc (avr-objcopy) / iccAVR (?)
  - Avr-gcc produit le mapping seul d'après le type de micro-contrôleur renseigné lors de la compilation
  - Avr-objcopy traduit le binaire (.elf) en format atmel , i.e. intel hexadecimal (.hex)

- AVRProg (version windows)
  - Utilise le port série pour communiquer avec le micro-contrôleur
  - Écrit soit en Flash, soit dans l'EEPROM
  - Peut facilement être utilisé avec Wine
  - *d'autres sont plus complet mais plus difficile à mettre en oeuvre selon les noyaux linux (**AvrDude**, uisp, AvrProg(linux) ...)*

Mise en oeuvre: Le code

- Code C classique
  - Arrêt des interruptions
  - **Configuration des registres**
  - Démarrage des interruptions
  - Boucle sans fin
- Forte utilisation des nombres hexadécimaux / binaire

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

binaire	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
hexa	**D**				**3**			
Décimal	**211**							

Mise en oeuvre: Le code

- Utilisation de types de données simples
  - Unsigned char = 8bits
  - Unsigned int = 16 bits
  - ➔ Les multiplications et divisions par des puissances de 2 se font par un simple décalage à gauche ou à droite

- Une page avec quelques rappels sympa :
  - http://wiki.jelectronique.com/avr-gcc

Travaux Pratiques

À vous de ~~jouer~~...

travailler