Microprocesseurs
&
Microcontrôleurs

Sylvain MONTAGNY

sylvain.montagny@univ-savoie.fr

Bâtiment chablais, bureau 13

04 79 75 86 86

Retrouver tous les documents de Cours/TD/TP sur le site
www.master-electronique.com

Présentation cours : Sommaire

- Cours : 10.5 h en 7 séances
  - Chapitre 1 : Rappels généraux sur les processeurs
  - Chapitre 2 : Les microcontrôleurs
    - Définition
    - Etude des périphériques
  - Chapitre 3 : La programmation

Présentation TD

- TD : 10.5 h en 7 séances
  - TD1 : Rappel sur les systèmes à microprocesseur. Cadencement d'un microcontrôleur. Instructions assembleurs.
  - TD2 : Utilisation du timer d'un microcontrôleur. Configuration des ports d'entrée/sortie d'un microcontrôleur.
  - TD3 : Les interruptions

Présentation TP

- TP : 12h en 3 séances de 4h
  - TP1 : Prise en main d'un environnement de programmation sur microcontrôleur
  - TP2 : Etude du Watchdog, et des interruptions dans un microcontrôleur
  - TP3 : Réalisation d'un minuteur à l'aide d'un afficheur 7 segment.

Chapitre 1 : Rappel généraux sur les processeurs

- 1.1 Rappel sur l'architecture interne des microprocesseurs
- 1.2 Le traitement des instructions
- 1.3 Les modes d'adressages
- 1.4 Exemple d'exécution d'un programme

L’architecture interne

Wafer

Un microprocesseur est constitué d’un morceau de silicium dopé. C’est donc un ensemble de millions de transistors.

- Wafer : Galette de plusieurs processeurs
- 1 processeur : quelques millimètres carrés

L’architecture interne

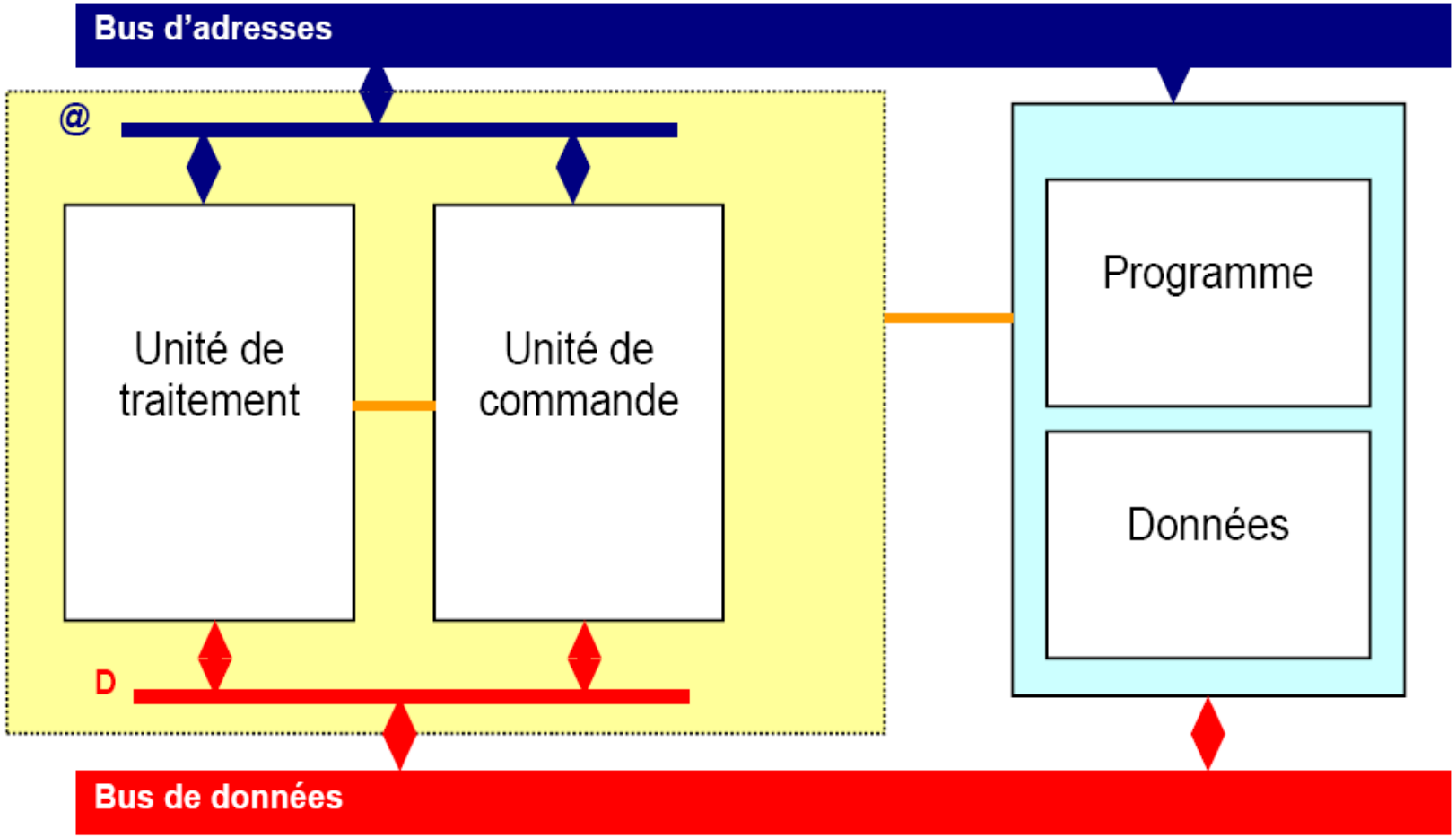
Unité commande/traitement

Un microprocesseur est construit autour de deux éléments principaux :

- Une unité de commande
- Une unité de traitement

L’architecture interne

Schéma



L’architecture interne

L’unité de commande (1)

Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire de l'instruction, le décodage de l’instruction codée sous forme binaire. Enfin elle pilote l’exécution de l’instruction.

Les blocs de l’unité de commande :

1. **Le compteur de programme (PC : Programme Counter) appelé aussi Compteur Ordinal (CO)** est constitué par un registre dont le contenu est initialisé avec l'adresse de la première instruction du programme. Il contient toujours l’adresse de la prochaine instruction à exécuter.

L’architecture interne

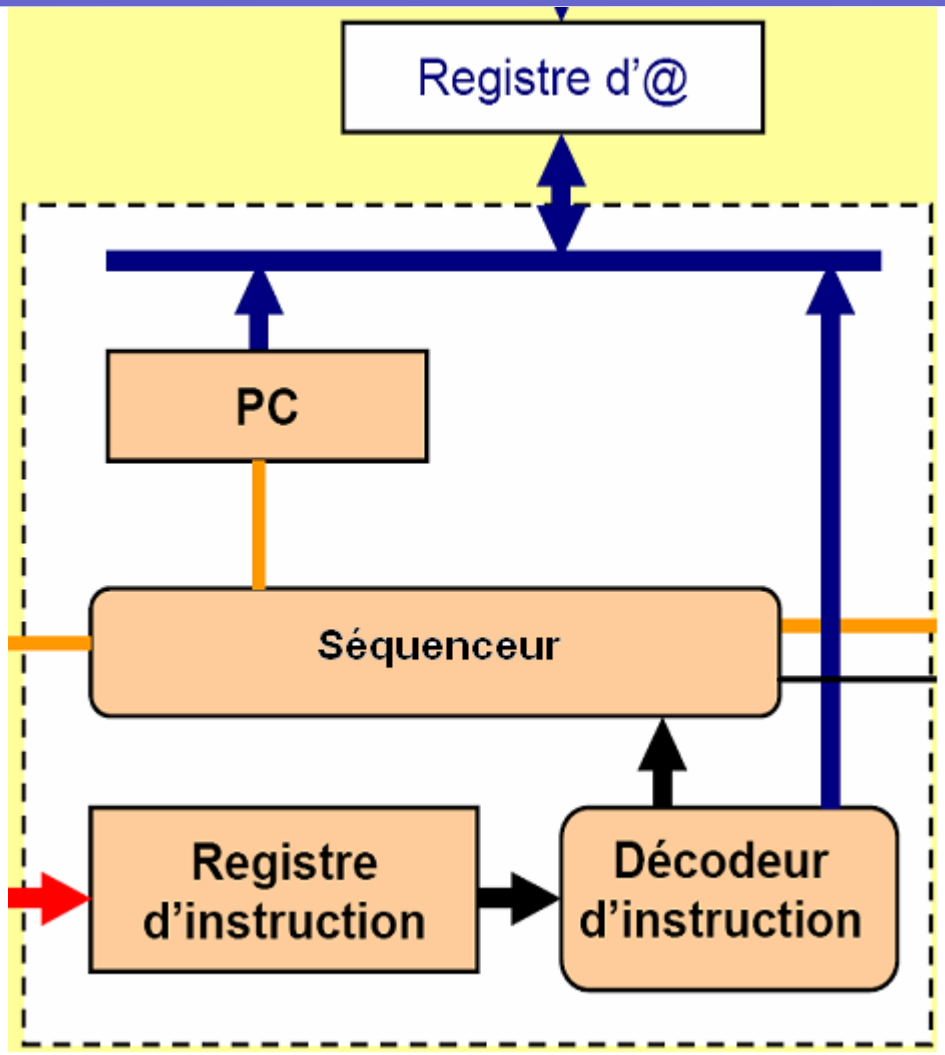
L’unité de commande (2)

2. **Le registre d'instruction et le décodeur d'instruction :** Chacune des instructions à exécuter est transféré depuis la mémoire dans le registre instruction puis est décodée par le décodeur d’instruction.

3. **Bloc logique de commande (ou séquenceur) :** Il organise l'exécution des instructions au rythme d’une horloge. Il élabore tous les signaux de synchronisation internes ou externes (bus de commande) du microprocesseur en fonction de l’instruction qu’il a a exécuter. Il s'agit d'un automate réalisé de façon micro-programmée.

L’architecture interne

L’unité de commande (3)



L’architecture interne

L’unité de traitement (1)

Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions

Les blocs de l’unité de traitement :

1. **Les accumulateurs** sont des registres de travail qui servent à stocker une opérande au début d'une opération arithmétique et le résultat à la fin de l'opération.
2. **L’Unité Arithmétique et Logique (UAL)** est un circuit complexe qui assure les fonctions logiques (ET, OU, Comparaison, Décalage, etc…) ou arithmétique (Addition, soustraction…).

L’architecture interne

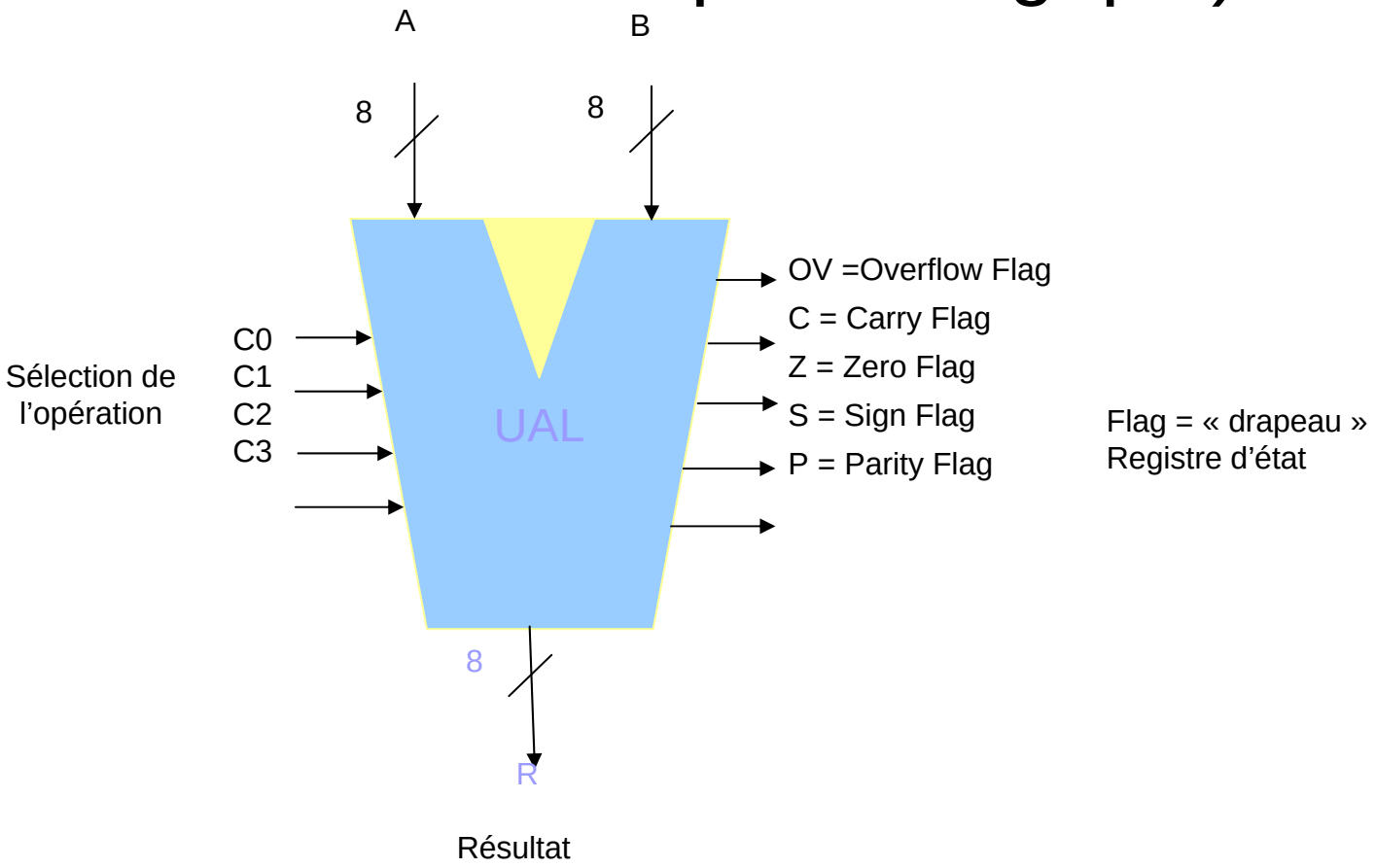
L’unité de traitement (2)

3. **Le registre d'état** est généralement composé de 8 bits à considérer individuellement. Chacun de ces bits est un indicateur dont l'état dépend du résultat de la dernière opération effectuée par l’UAL. On les appelle *indicateur d’état* ou *flag* ou *drapeaux*. Dans un programme le résultat du test de leur état conditionne souvent le déroulement de la suite du programme. On peut citer par exemple les indicateurs de :
   - Retenue **(carry : C)**
   - Débordement **(overflow : OV ou V)**
   - Zéro **(Z)**
   - ...

L’architecture interne

L’Unité de traitement (3)

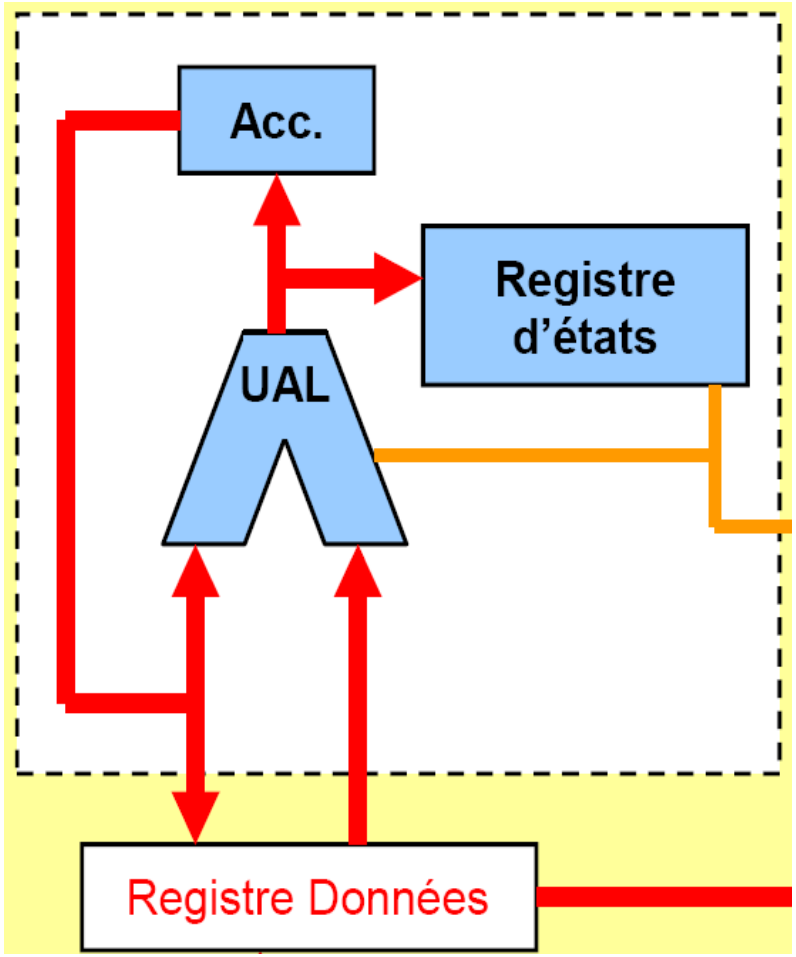
UAL : Unité Arithmétique et Logique)



Flag = « drapeau »
Registre d’état

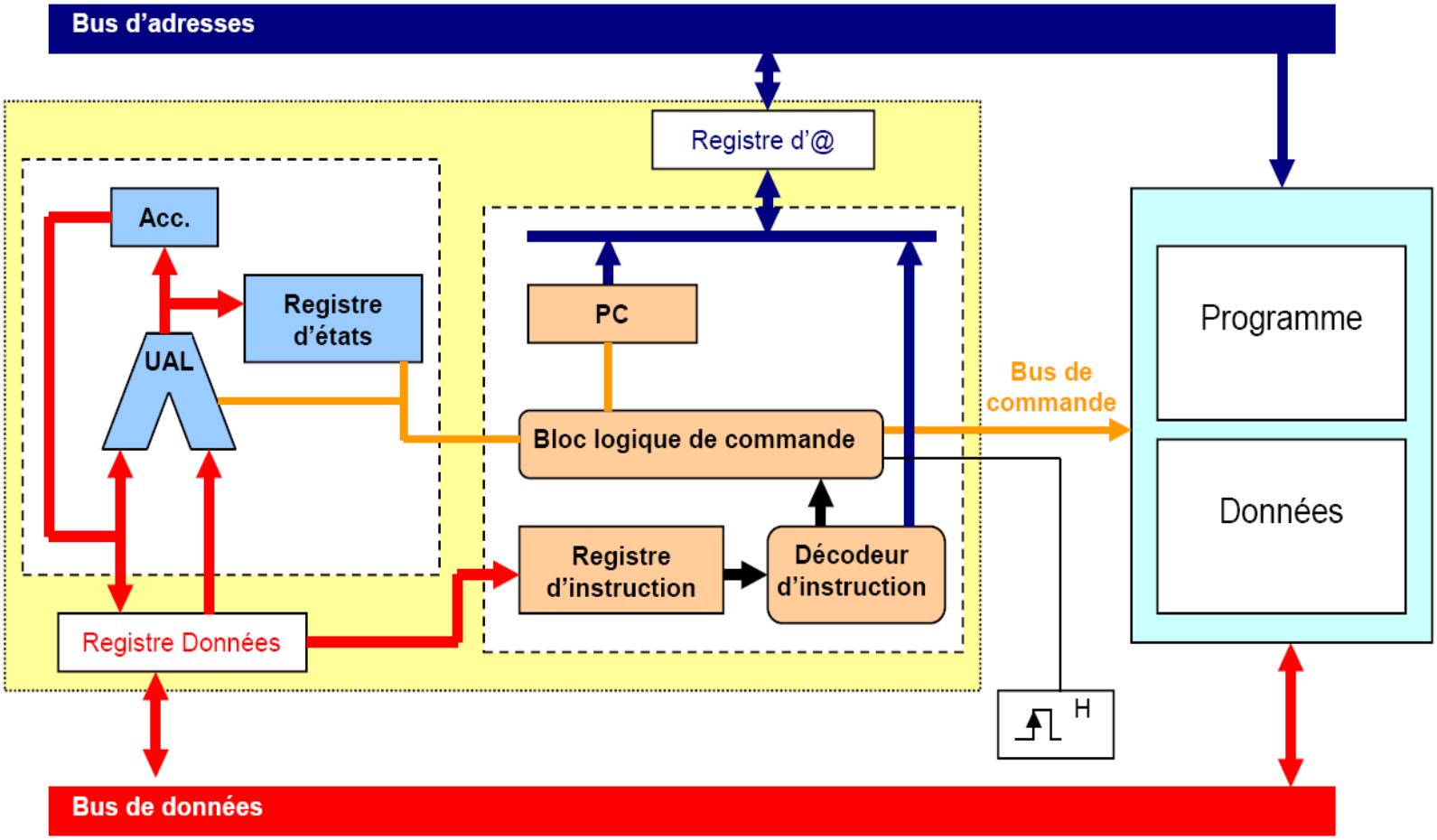
L’architecture interne

L’unité de traitement (4)

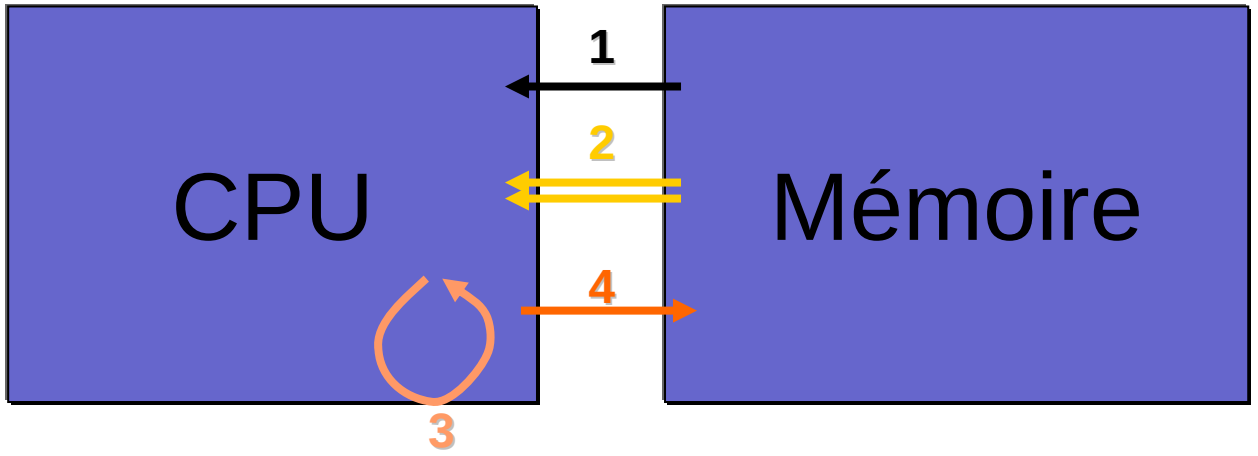


L'architecture interne

Architecture complète

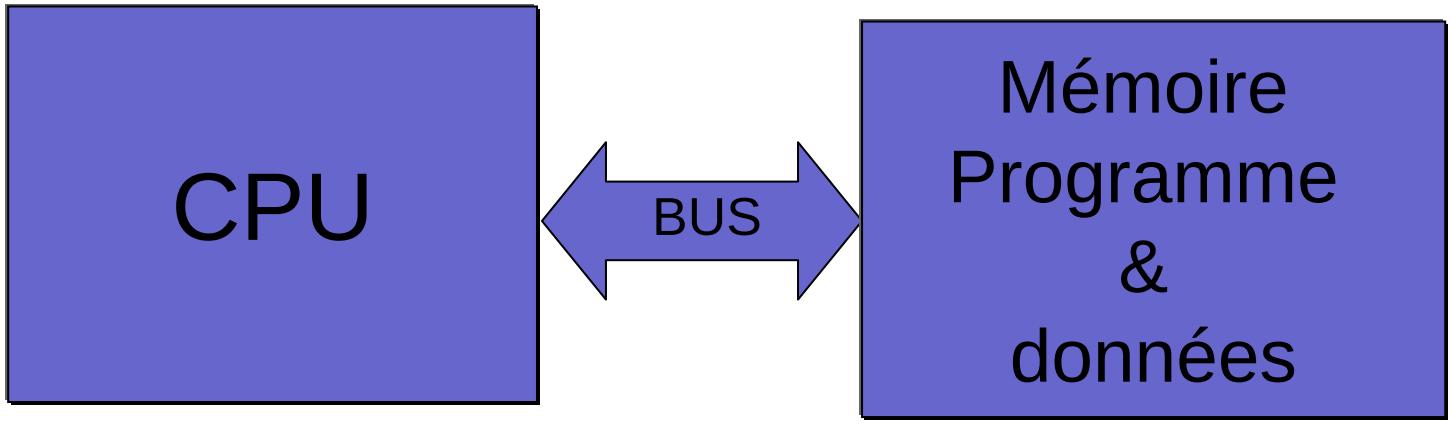


Rappels: le fonctionnement basique d'une opération de calcul



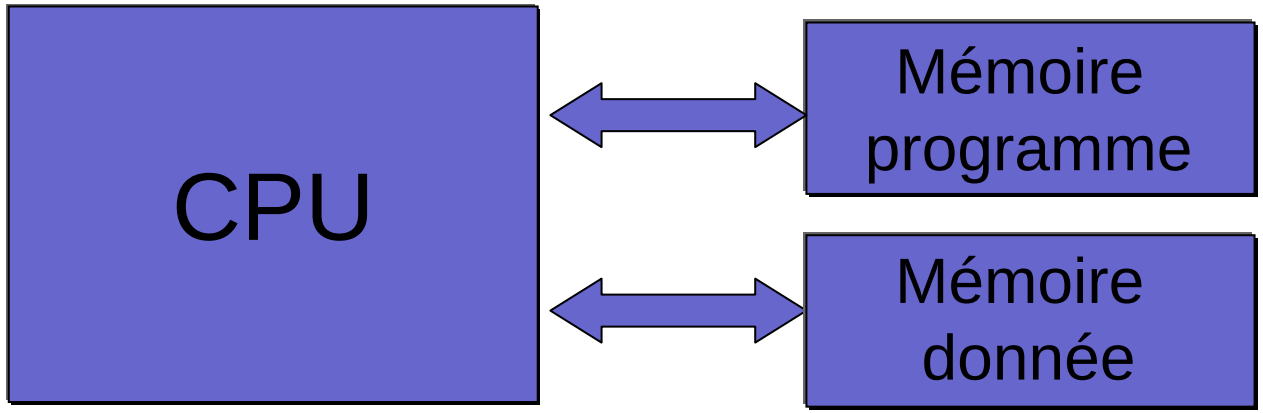
- (1) Charger une instruction depuis la mémoire
- (2) Charger les opérandes depuis la mémoire
- (3) Effectuer les calculs
- (4) Stocker le résultat en mémoire

L’architecture
Von Neuman



- Un seul chemin d'accès à la mémoire
  - Un bus de données (programme et données),
  - Un bus d’adresse (programme et données)
- Architecture des processeurs d’usage général
- Goulot d'étranglement pour l'accès à la mémoire

L'architecture
Harvard

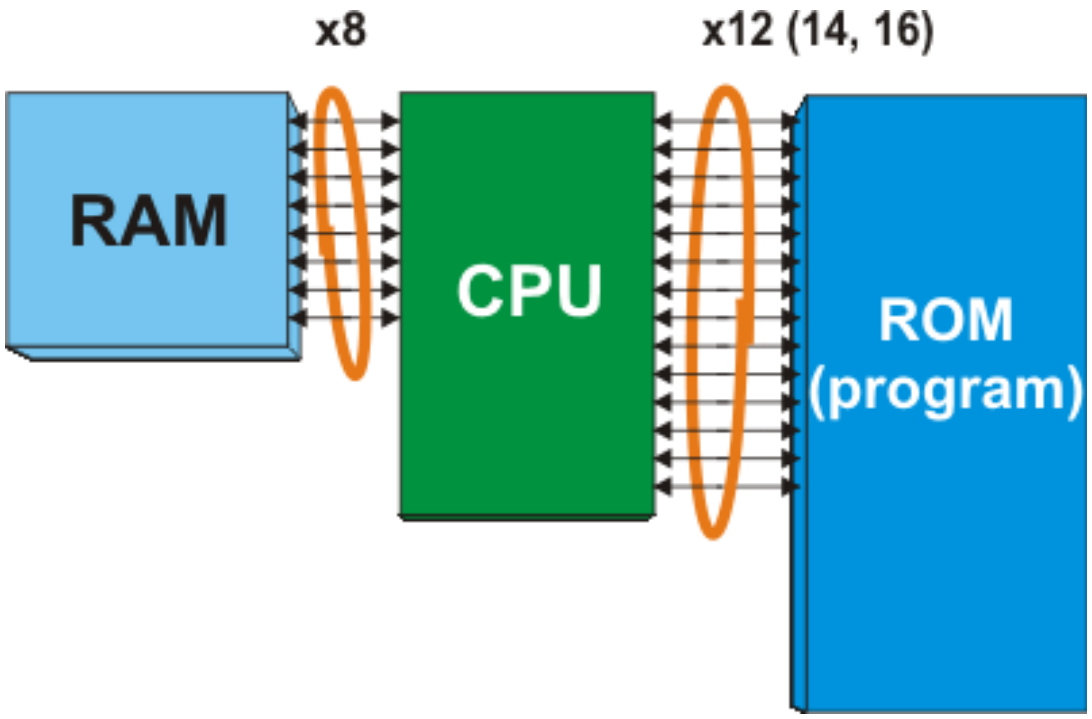


- Séparation des mémoires programme et données
  - Un bus de données programme,
  - Un bus de données pour les données,
  - Un bus d'adresse programme,
  - Un bus d'adresse pour les données.
- Meilleure utilisation du CPU :
  - Chargement du programme et des données en parallèle

L’architecture

Harvard : Cas des microcontrôleurs PIC

- Seul les bus de donnée (data ou instructions) sont représentées



Chapitre 1 : Rappel généraux sur les processeurs

- 1.1 Rappel sur l'architecture interne des microprocesseurs
- 1.2 Le traitement des instructions
- 1.3 Les modes d'adressages
- 1.4 Exemple d'exécution d'un programme

Le traitement des instructions

Organisation d’une instruction

Le microprocesseur ne comprend qu’un certain nombre d’instructions qui 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.

Le traitement des instructions
Exemple d'instruction

- Instruction Addition :

  Accumulateur = Accumulateur + Opérande

Correspond à l'instruction ADD A,#2

Instruction (16 bits)	
Code opératoire (5 bits)	Champ opérande (11 bits)
ADD A	#2
11001	000 0000 0010

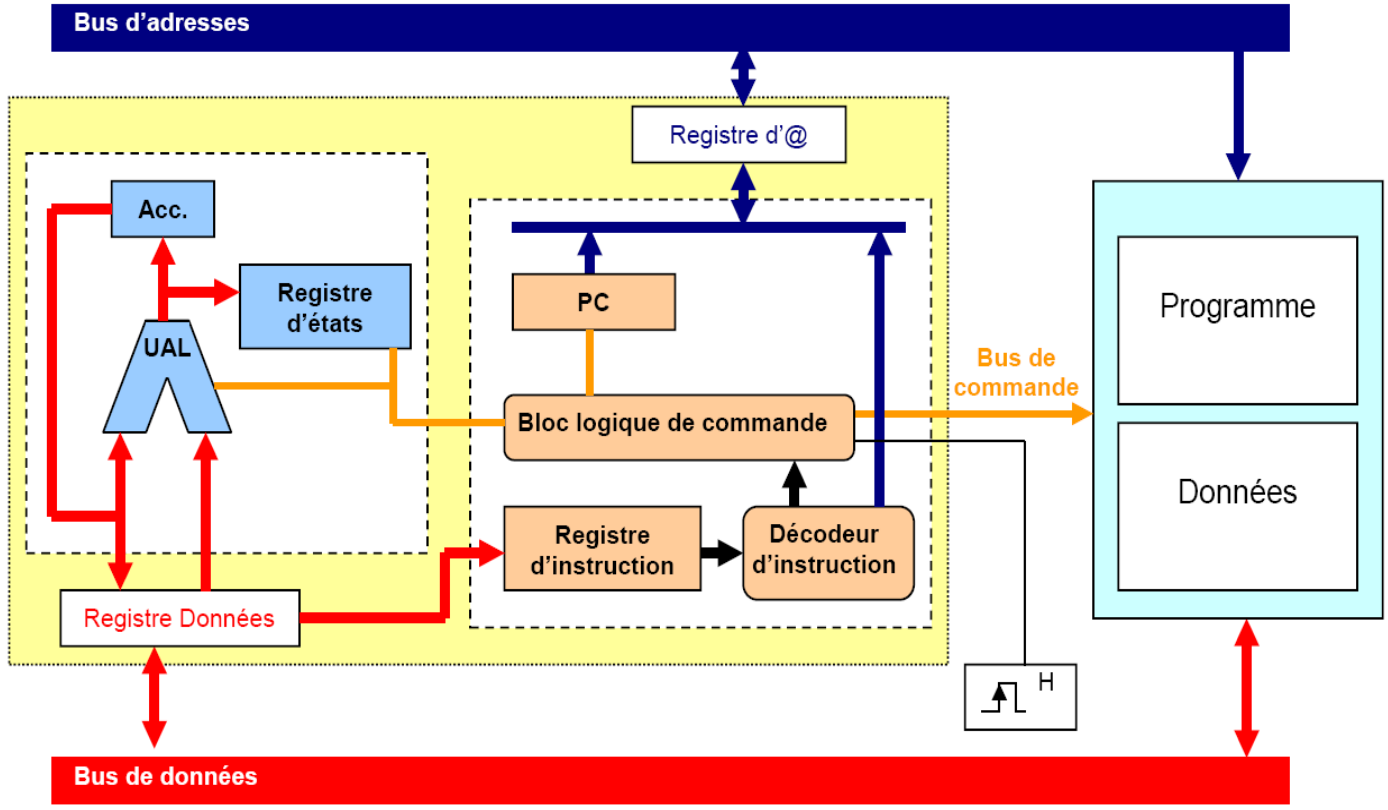
Cette instruction est comprise par le processeur par le mot binaire :

11001 000 0000 0010 = code machine

Le traitement des instructions

Phase 1 : Recherche de l'instruction en mémoire

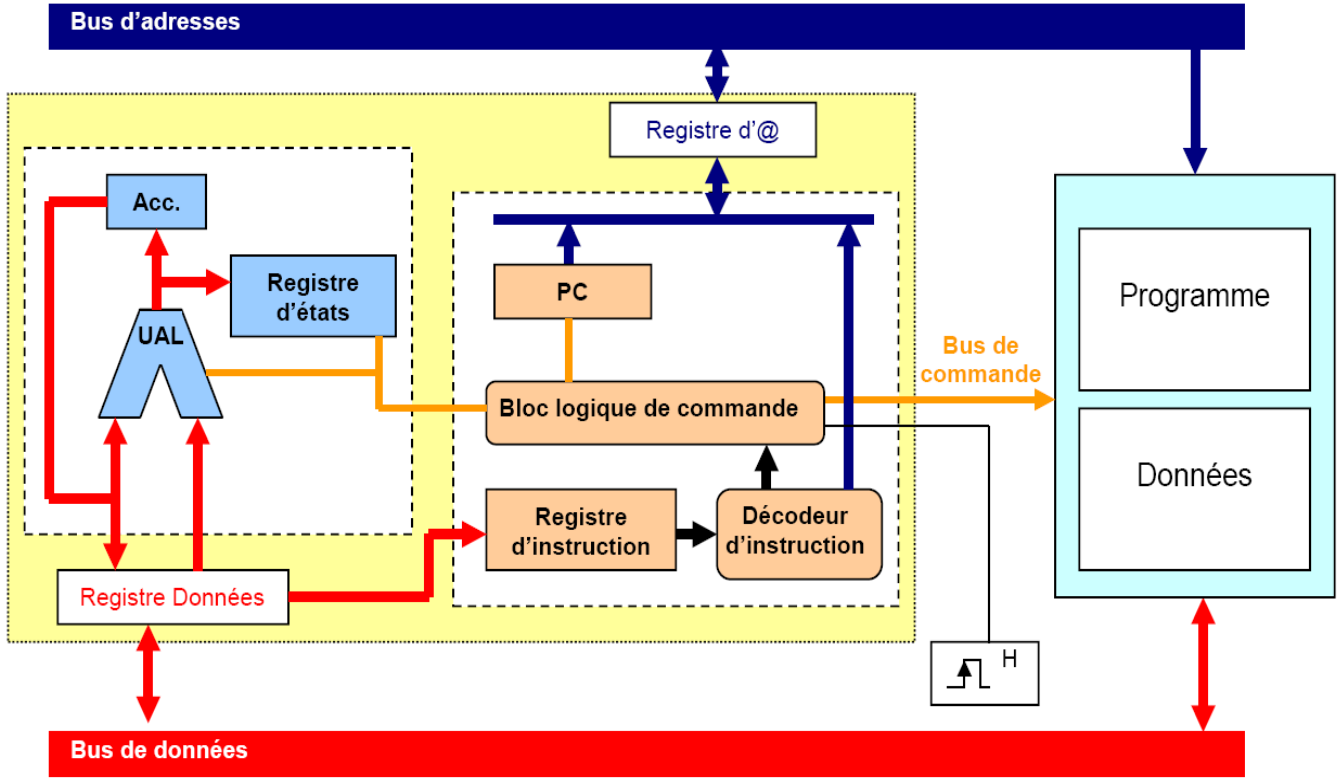
- La valeur du PC est placée sur le bus d'adresse par l'unité de commande qui émet un ordre de lecture.
- Après le temps d'accès à la mémoire, le contenu de la case mémoire sélectionnée est disponible sur le bus des données.
- L'instruction est stockée dans le registre d'instruction du processeur.



Le traitement des instructions

Phase 2 : Décodage et recherche de l'opérande

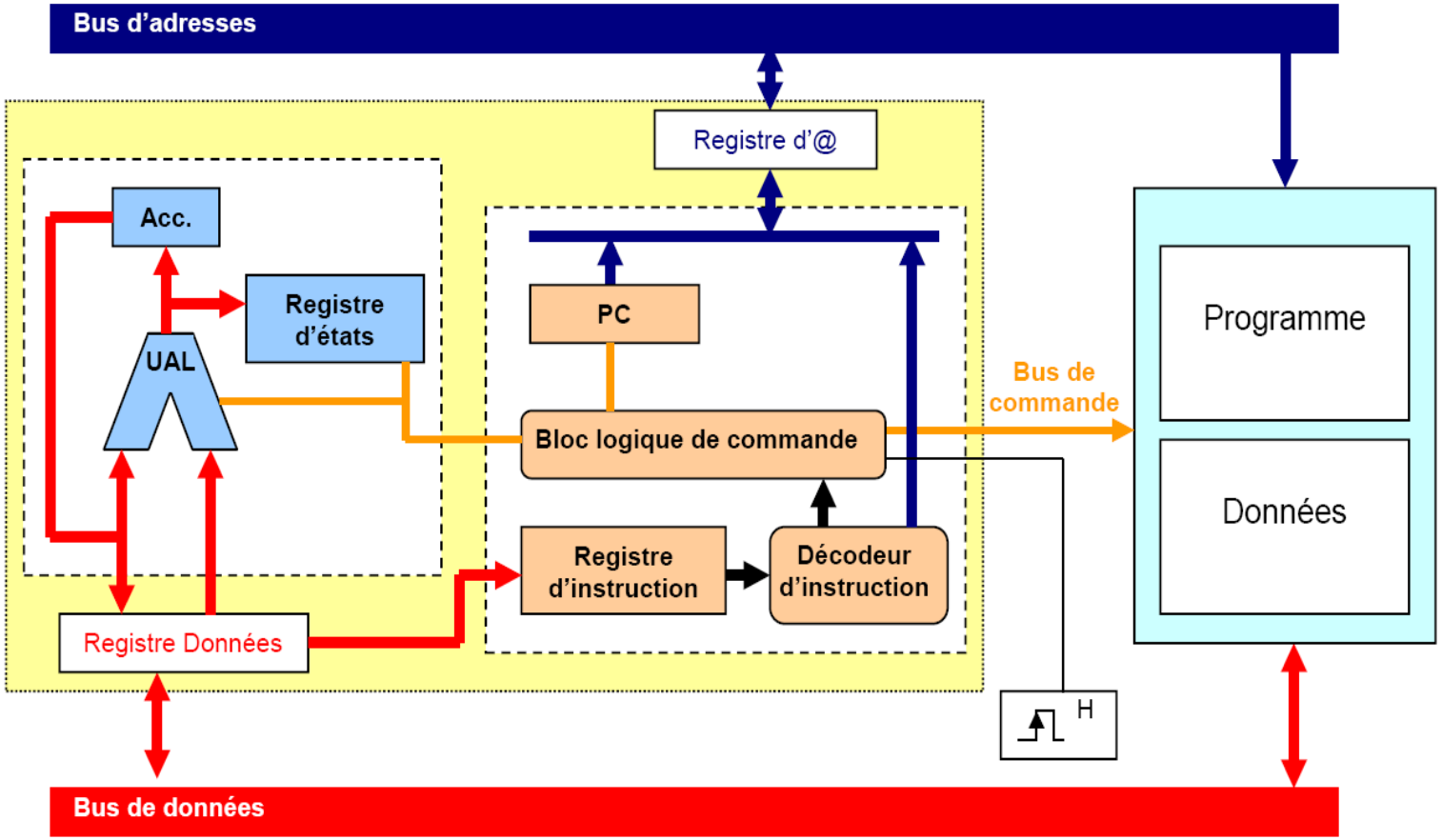
- L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.
- Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur sur le bus de données.
- L'opérande est stocké dans le registre de données.



Le traitement des instructions

Phase 3 : Exécution de l'instruction

- Le séquenceur réalise l'instruction.
- Les drapeaux sont positionnés (registre d'état).
- L'unité de commande positionne le PC pour l'instruction suivante.



Le traitement des instructions
Les architectures RISC et CISC (1)

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- L' architecture CISC

  (Complex Instruction Set Computer)

- L'architecture RISC

  (Reduced Instruction Set Computer)

Le traitement des instructions

Les architectures RISC et CISC (2)

Architecture RISC	Architecture CISC
instructions simples ne prenant qu'un seul cycle	instructions complexes prenant plusieurs cycles
instructions au format fixe	instructions au format variable
décodeur simple (câblé)	décodeur complexe (microcode)
beaucoup de registres	peu de registres
peu de modes d'adressage	beaucoup de modes d'adressage
compilateur complexe	compilateur simple

Chapitre 1 : Rappels généraux sur les processeurs

- 1.1 Rappel sur l'architecture interne des microprocesseurs
- 1.2 Le traitement des instructions
- 1.3 Les modes d'adressages
- 1.4 Exemple d'exécution d'un programme

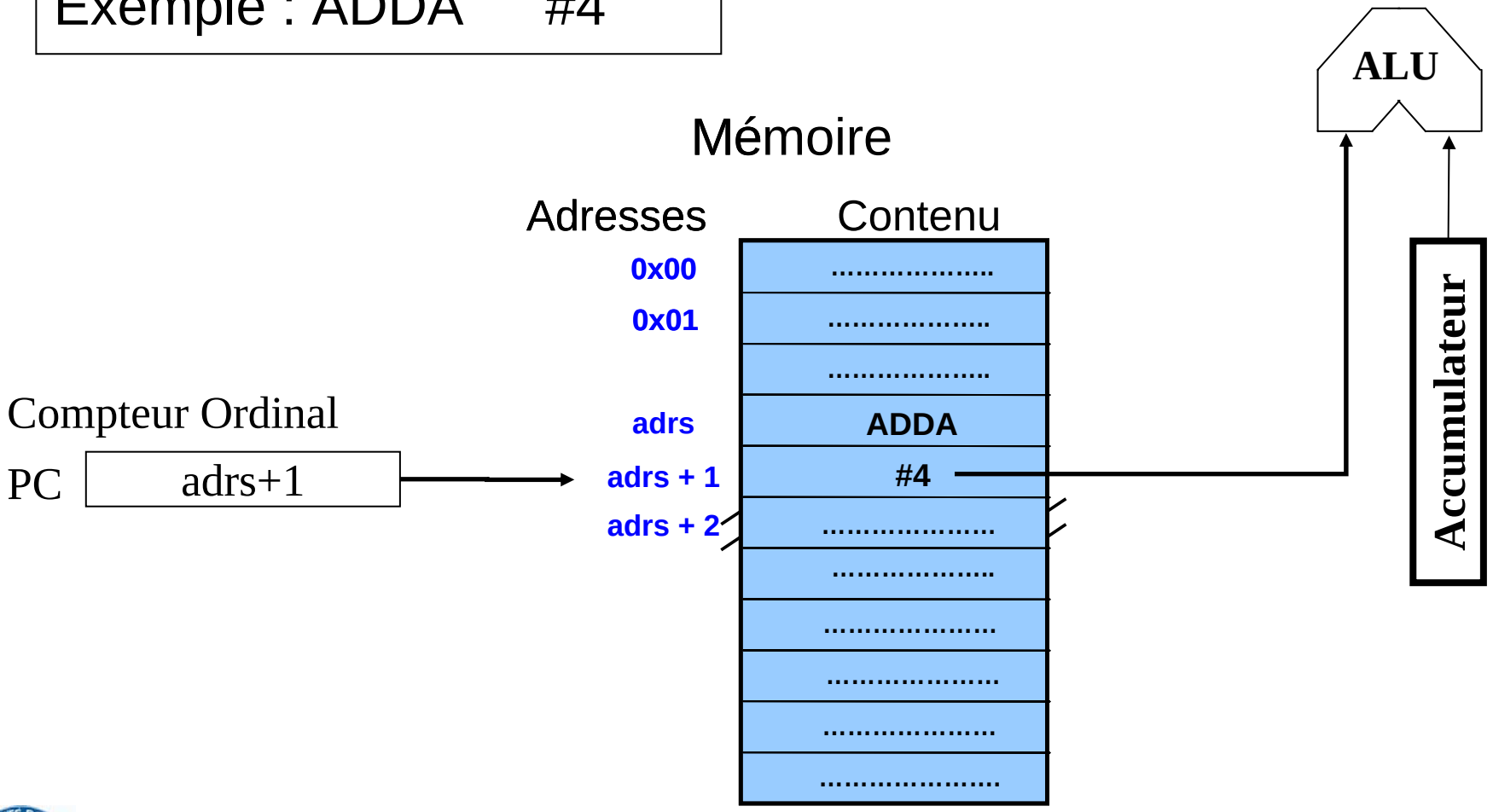
Les modes d’adressages

- Ce sont les diverses manières de définir la localisation d ’un opérande. Les trois modes d’adressage les plus courant sont :
  - Adressage immédiat
  - Adressage direct
  - Adressage indirect

Les modes d'adressages

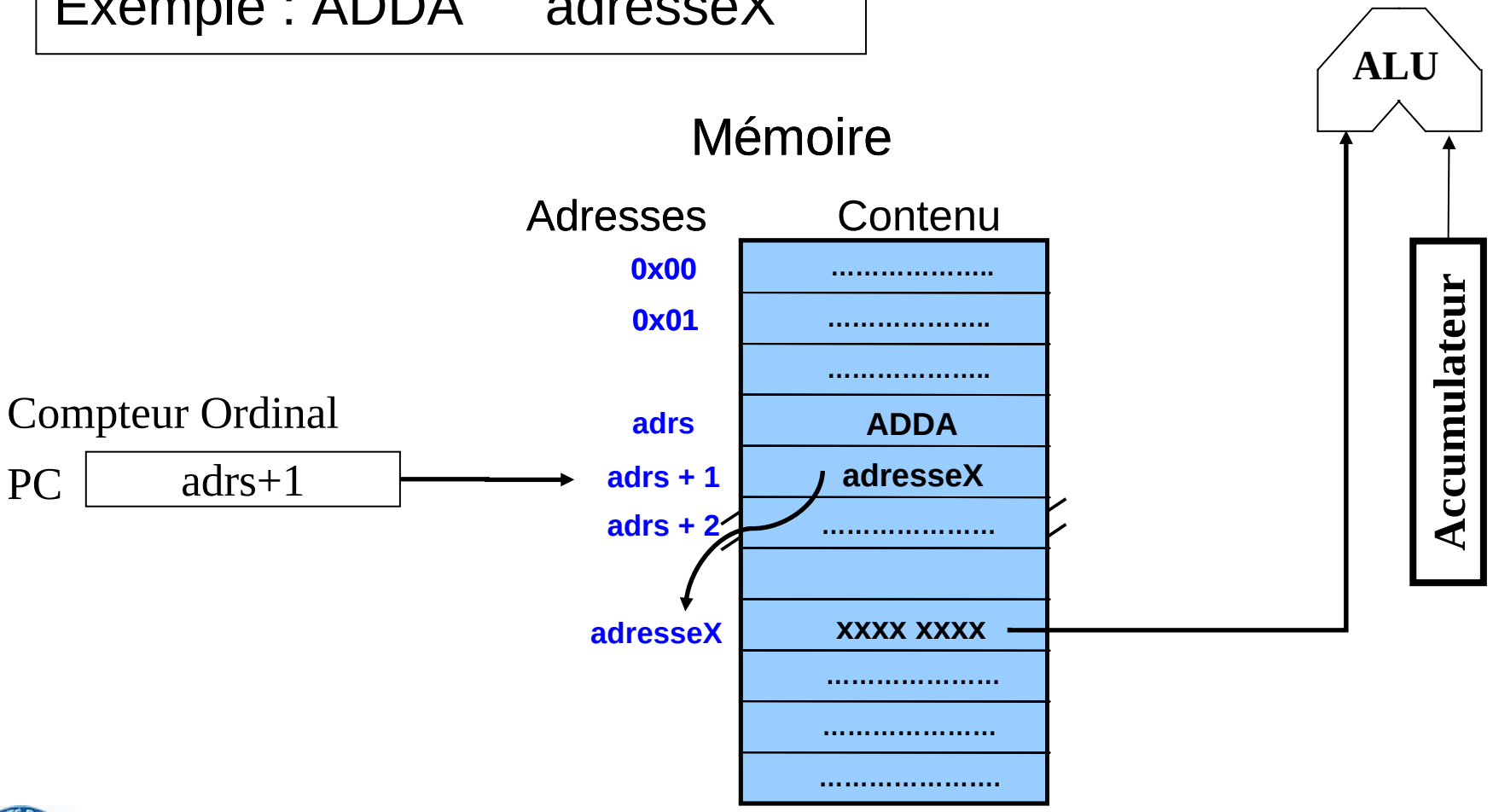
Immédiat

Exemple : ADDA #4



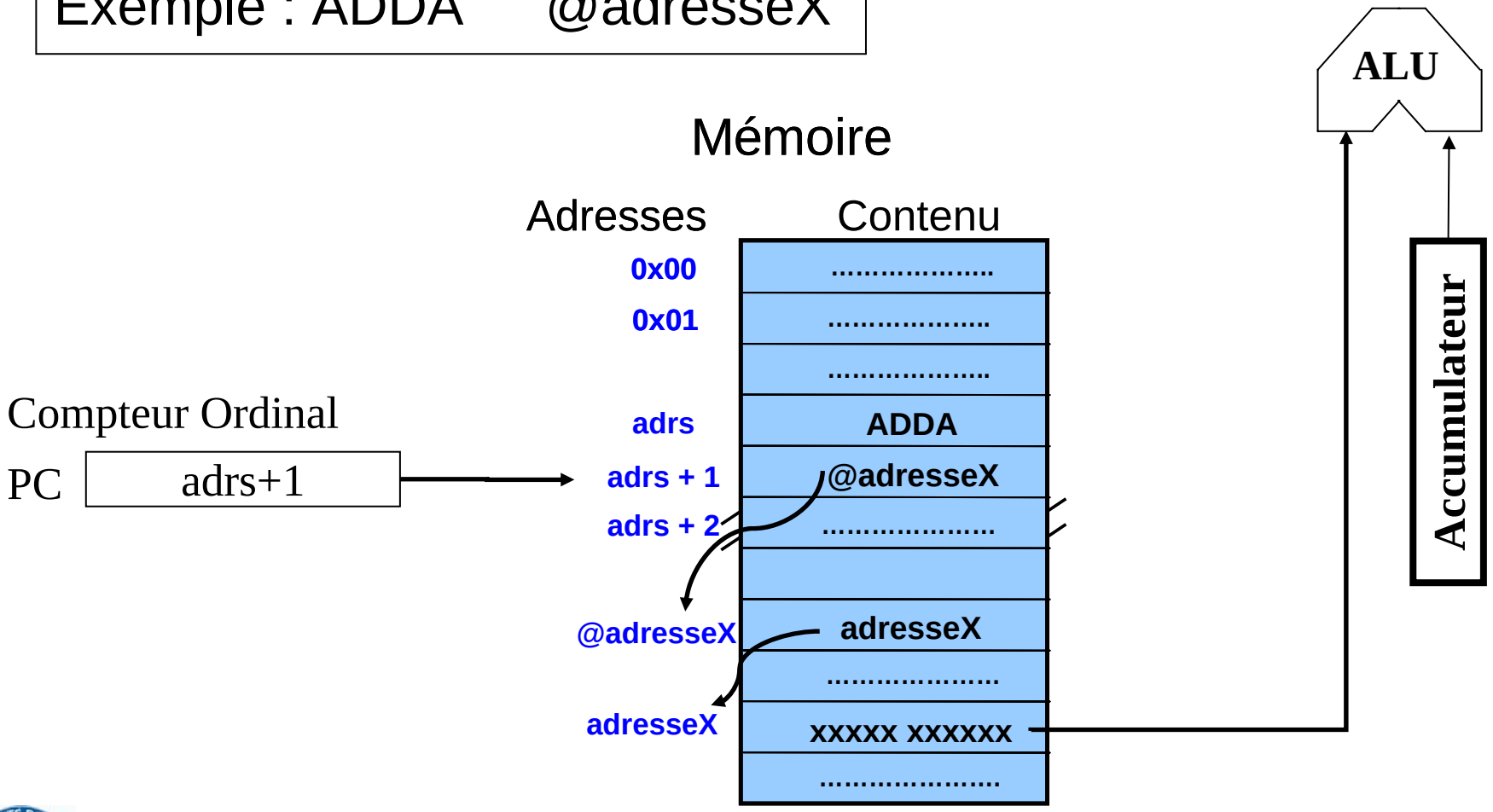
Les modes d'adressages
Direct

Exemple : ADDA adresseX



Les modes d'adressages
Indirect

Exemple : ADDA @adresseX



Les modes d’adressages

- Pourquoi existe-t-il plusieurs modes d’adressage ?

Chapitre 1 : Rappel généraux sur les processeurs

- 1.1 Rappel sur l'architecture interne des microprocesseurs
- 1.2 Le traitement des instructions
- 1.3 Les modes d'adressages
- 1.4 Exemple d'exécution d'un programme

Exemple d'exécution

Directives d'assemblage

Valeurs des symboles

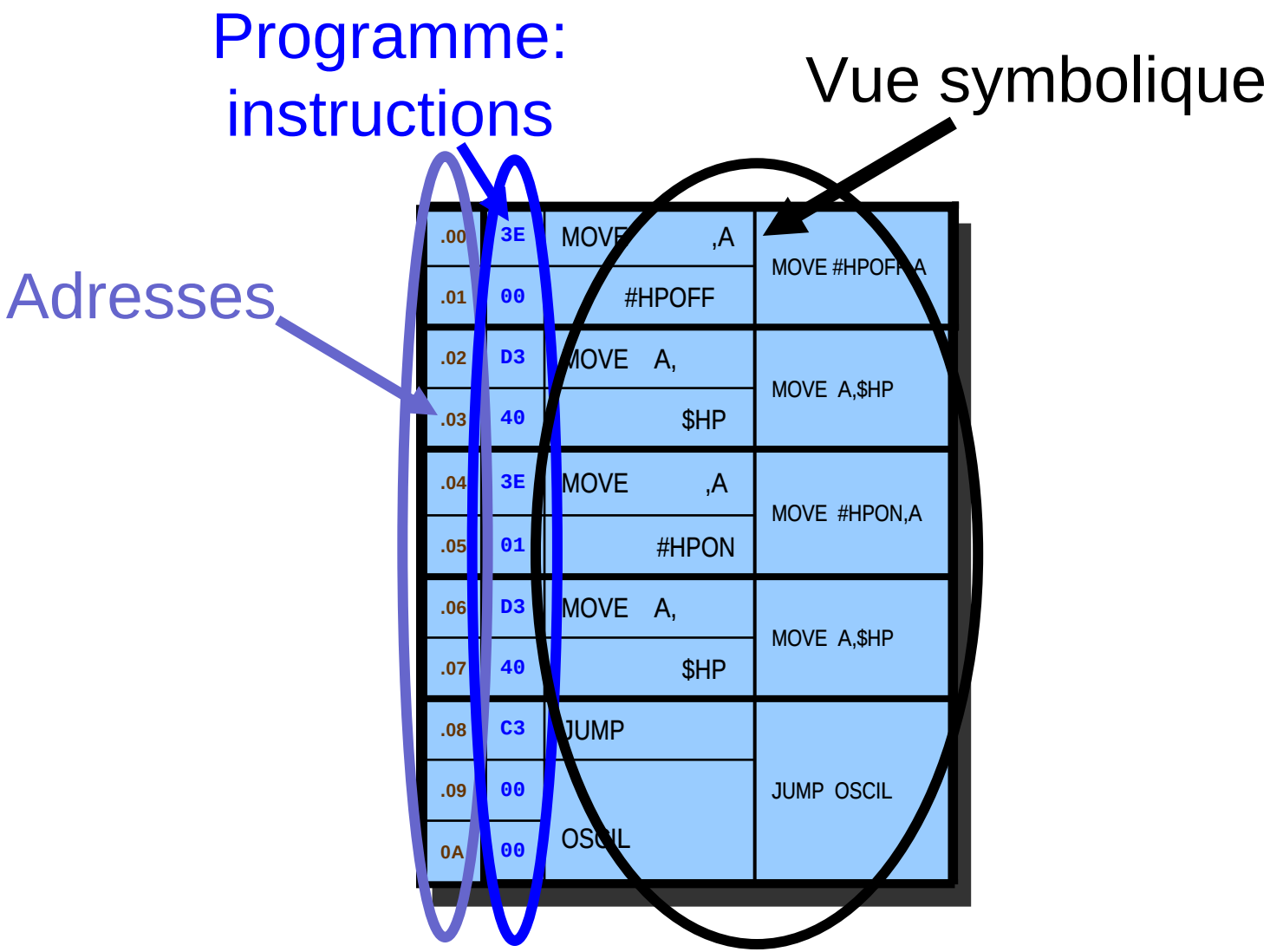
```
                .TITLE    Bruit_HP      ; Titre du programme
                .PROC     I8085         ; Processeur utilisé
                .START    OSCIL         ; Adresse début programme

00000040  HP      =   10'64     ; Adresse  du Haut-Parleur (40 Hexa)
00000000  HPOFF   =   0         ; Constante, membrane relachée
00000001  HPON    =   1         ; Constante, membrane attirée

000000          .LOC 0          ; Adresse d'assemblage du programme

000000  OSCIL:
```

Mnémoniques des instructions

Adresses	Code des instructions		Commentaires
000000	3E 00	MOVE #HPOFF, A	; Charge valeur HPOFF (0) dans l'accumulateur A
000002	D3 40	MOVE A, $HP	; Charge A sur périphérique HP
000004	3E 01	MOVE #HPON, A	; Charge valeur HPON (1) dans l'accumulateur A
000006	D3 40	MOVE A, $HP	; Charge A sur périphérique HP
000008	C3 00 00	JUMP OSCIL	; Saute au début à OSCIL
00000B			

```
                .END            ; Fin de l'assemblage
```

Exemple d'exécution



Exemple d’exécution

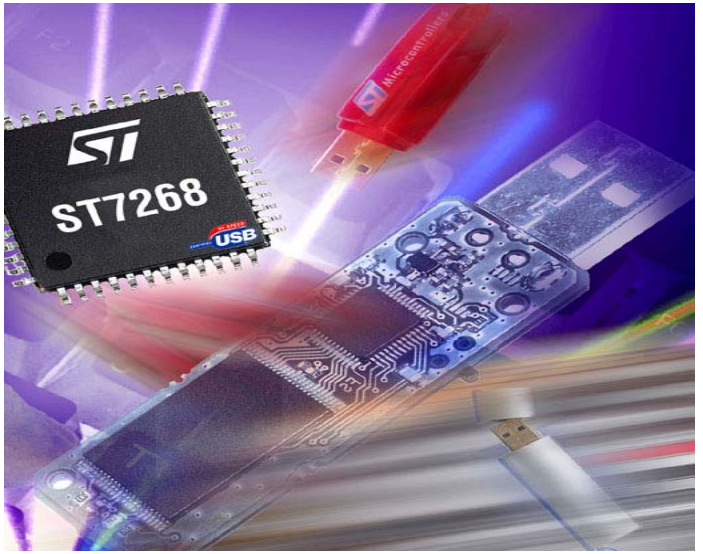
Continue….

1ière boucle

2ième boucle

3ième boucle

4ième boucle

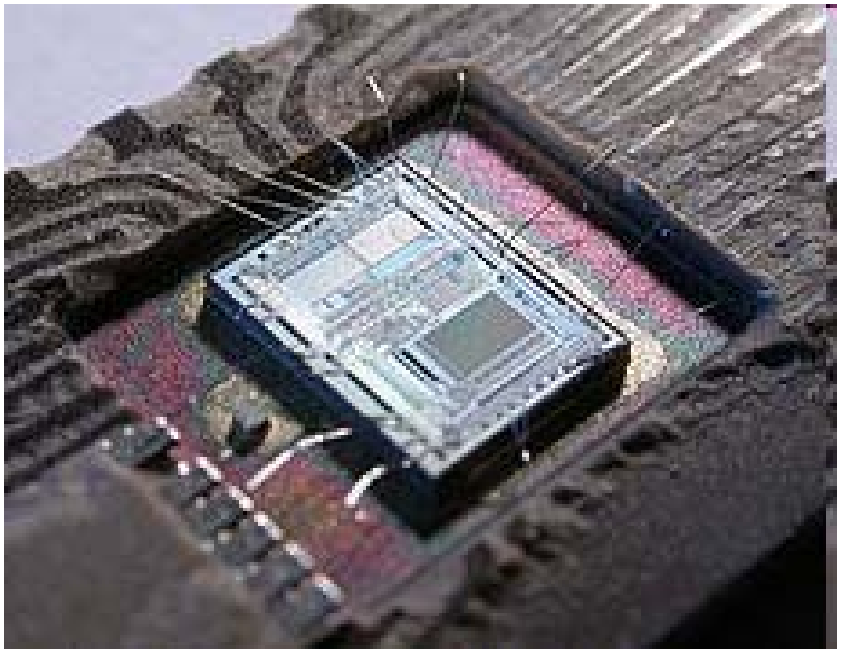


Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

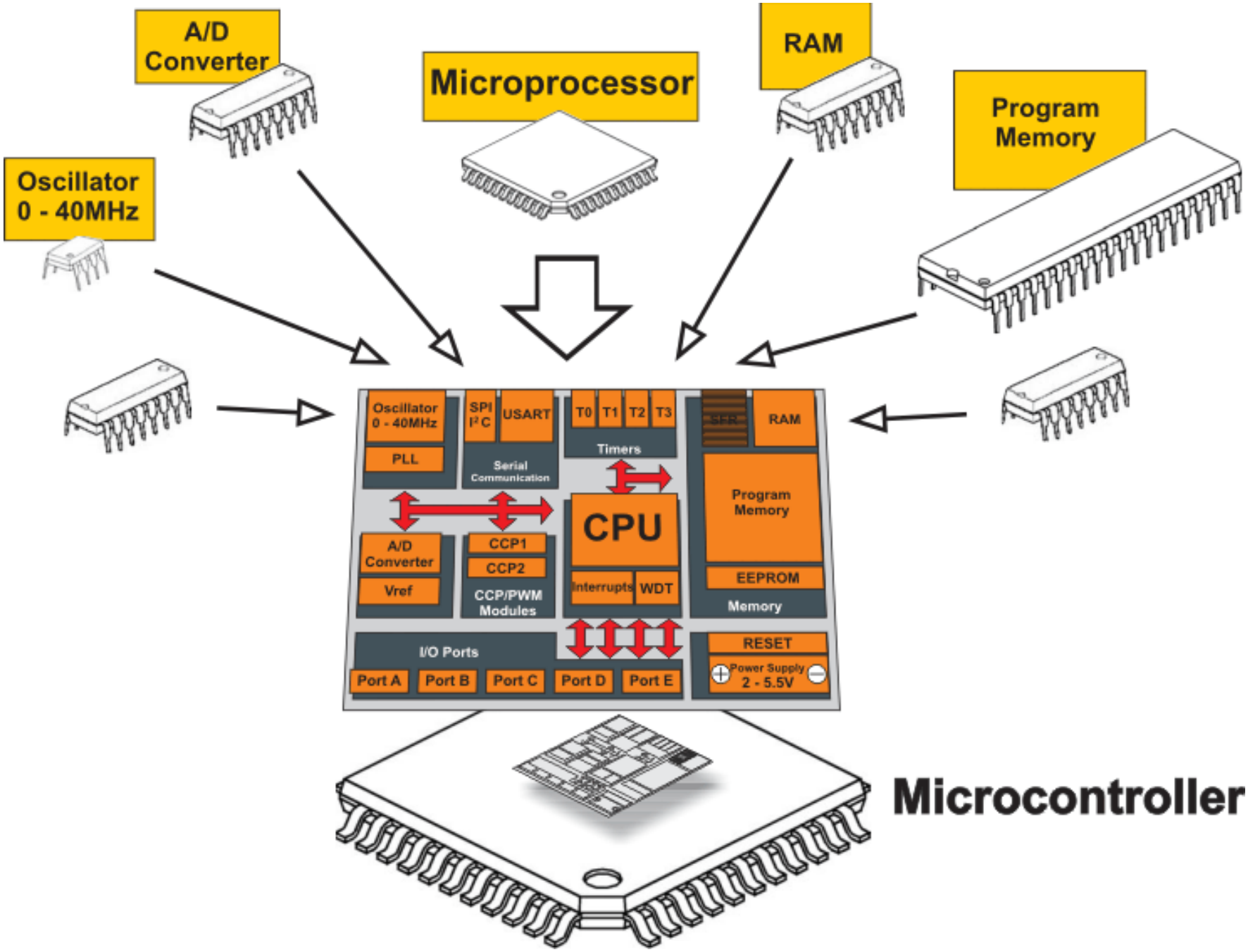
Définition d’un microcontrôleur

Un microcontrôleur est un circuit qui intègre un maximum de fonctions dans un même boitier. L’intégration de ces fonctions dans le même environnement permet de créer des applications plus simplement.



Le circuit intégré d'un microcontrôleur 8 bits Intel 8742 possède sur une unique puce :

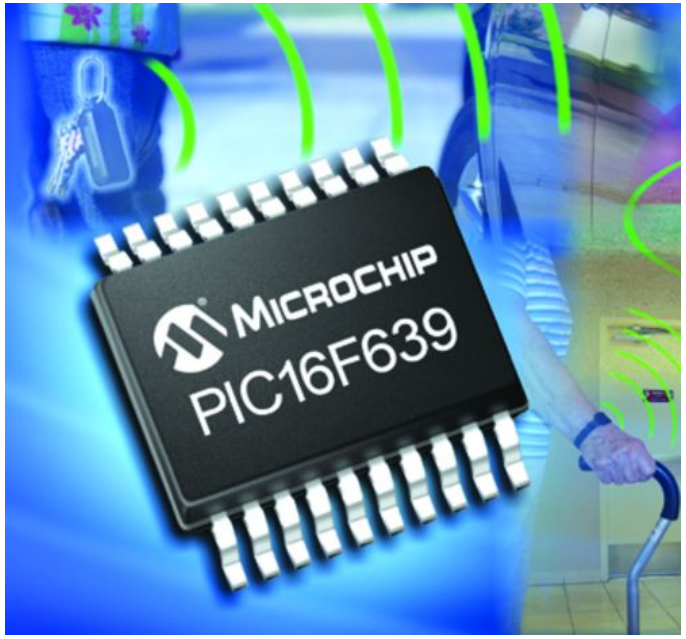
- Un processeur cadencé à 12 MHz
- 128 octets de mémoire vive
- Une EPROM de 2048 bits
- De nombreuse entrées-sorties


A/D Converter
Microprocessor
RAM
Oscillator 0 - 40MHz
Program Memory
Oscillator 0 - 40MHz
PLL
SPI I²C
USART
Serial Communication
T0
T1
T2
T3
Timers
SFR
RAM
CPU
Program Memory
A/D Converter
Vref
CCP1
CCP2
CCP/PWM Modules
Interrupts
WDT
EEPROM
Memory
I/O Ports
Port A
Port B
Port C
Port D
Port E
RESET
Power Supply
2 - 5.5V
Microcontroller

Définition d'un microcontrôleur

Avantages

- Cout réduit
- Encombrement moindre
- Fiabilité
- Mise en œuvre plus simple
- Consommation plus faible



Définition d'un microcontrôleur

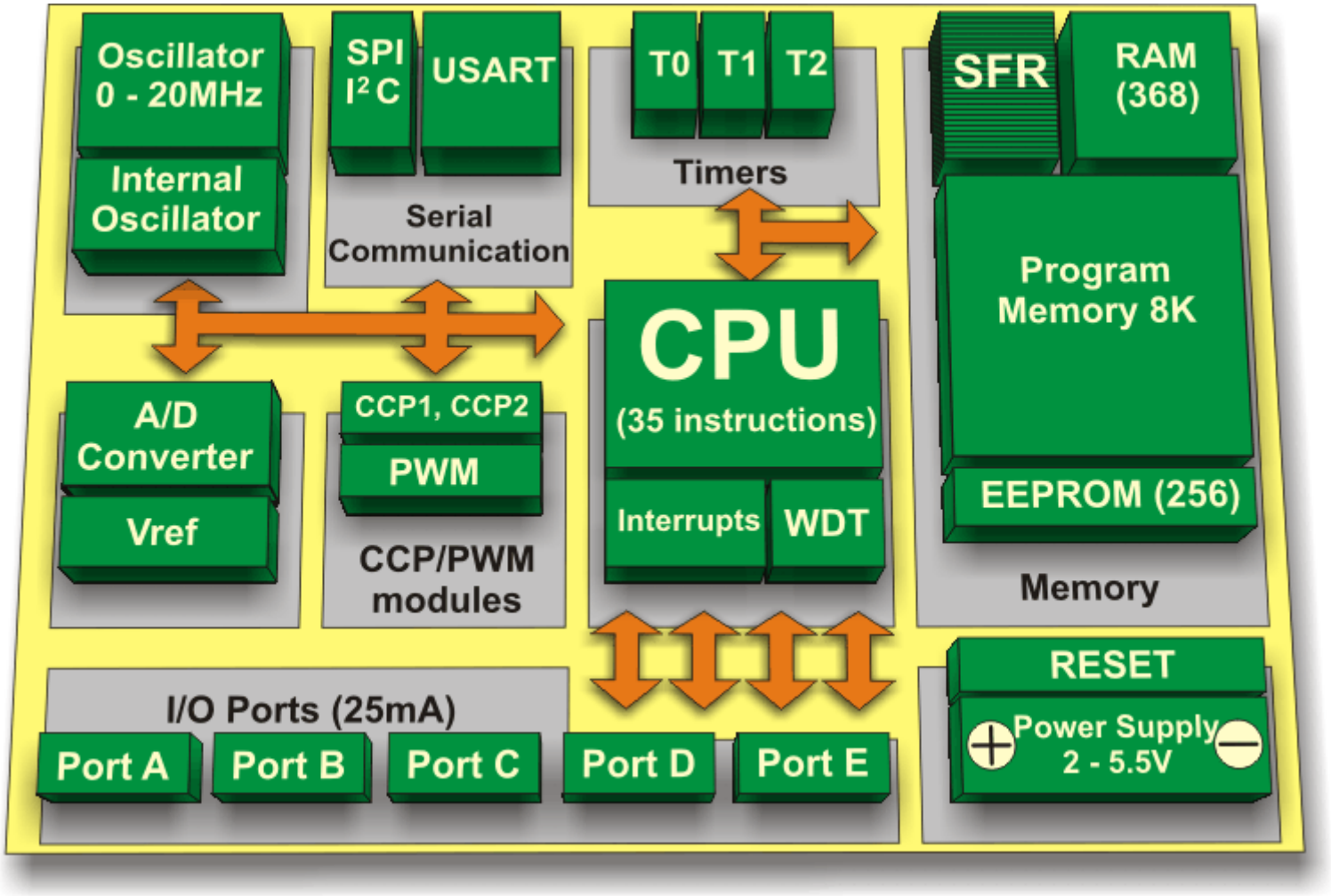
Contenu

- La structure interne d'un microcontrôleur comporte typiquement :
  - Une unité de calcul et de commande
  - Mémoire ROM
  - Mémoire RAM
  - Un contrôleur d'interruption
  - Un compteur/temporisateur (timer)
  - Des entrées/sorties parallèles (ports)
  - Un UART (port série)

- Il peut aussi posséder :
  - Un Watchdog : (surveillance du programme)
  - Une sortie PWM (modulation d'impulsion)
  - Un CAN/CNA (Convertisseur analogique numérique)
  - Un interface I^2C, CAN…

Définition d'un microcontrôleur

Exemple : Microcontrôleur PIC 16F877

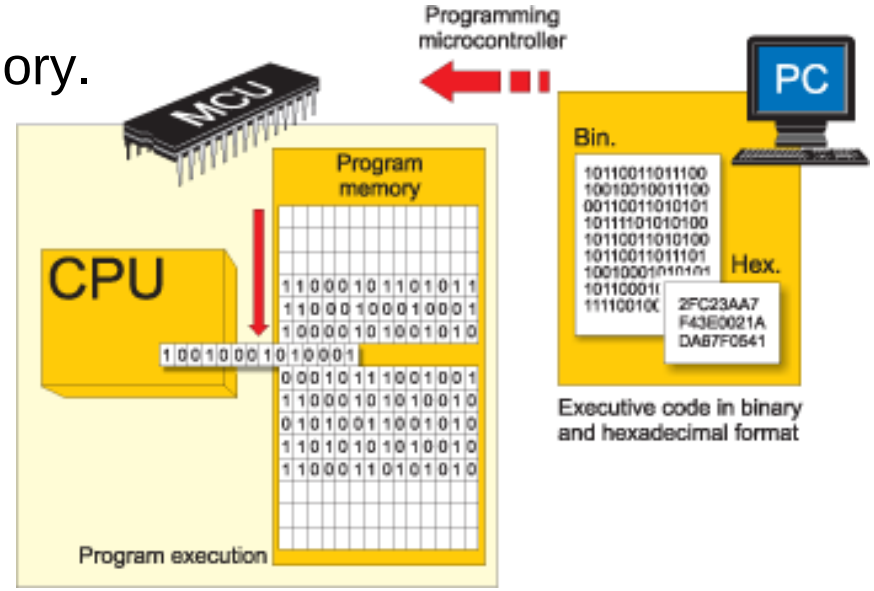


Définition d'un microcontrôleur

Les mémoires

- ROM memory :

  Aussi appelé (à juste titre) program memory. C'est une mémoire Flash qui contient le programme à exécuter.



- EEPROM memory

  C'est une mémoire similaire à la mémoire programme. En revanche, le contenu peut être modifié en cours d'utilisation de l'application.

Définition d'un microcontrôleur

Les mémoires

- RAM memory :
  - **General Purpose Register :** Mémoire RAM classique, utiliser pour stocké des variables. Exemple :

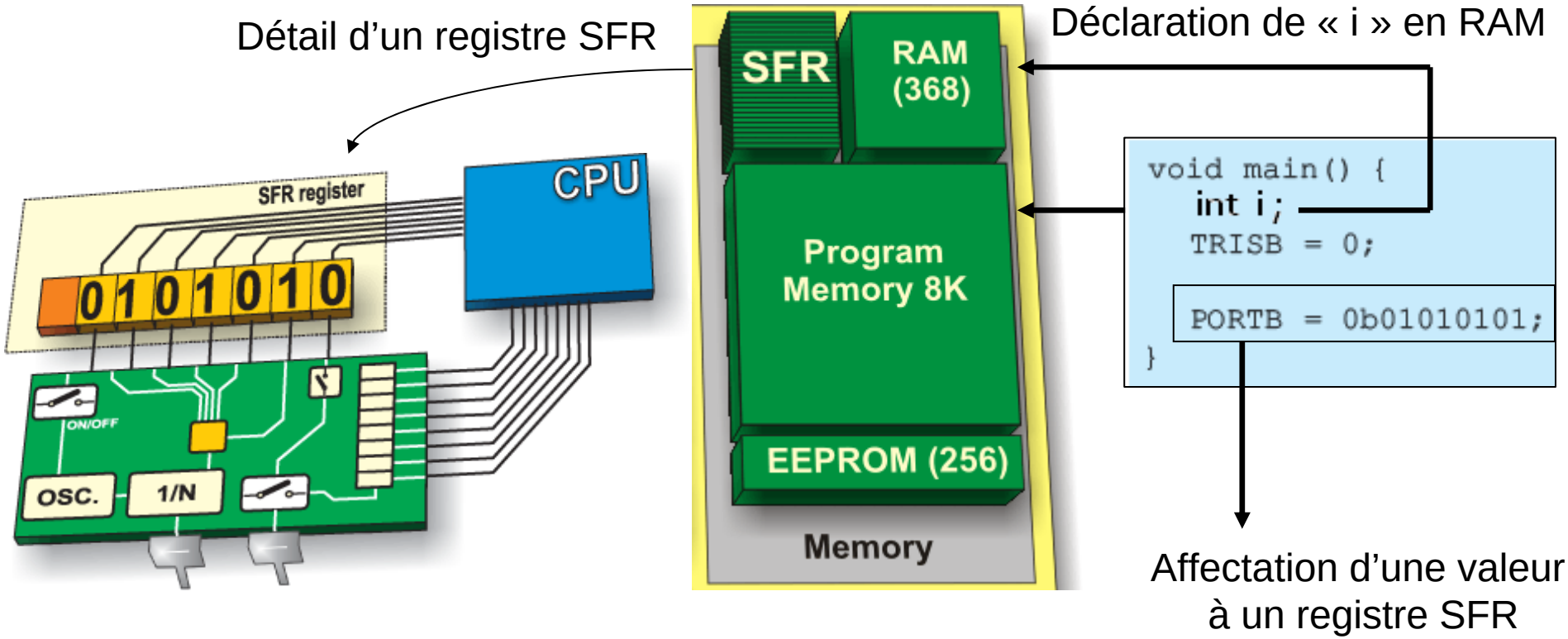
```
int i;
i++;            // incrémentation de i depuis la RAM
```

  - **SFR (Special Function Register) :** C'est aussi de la mémoire RAM, sauf que les rôles de chacune des cases mémoire (registres) ont été définis par le fabriquant. Chaque registre SFR est connecté à un périphérique matériel spécifique et permet de la contrôler. Exemple :

ADCON0 register (adresse 9Fh) permet de piloter le convertisseur A/D.

Définition d'un microcontrôleur

Les mémoires

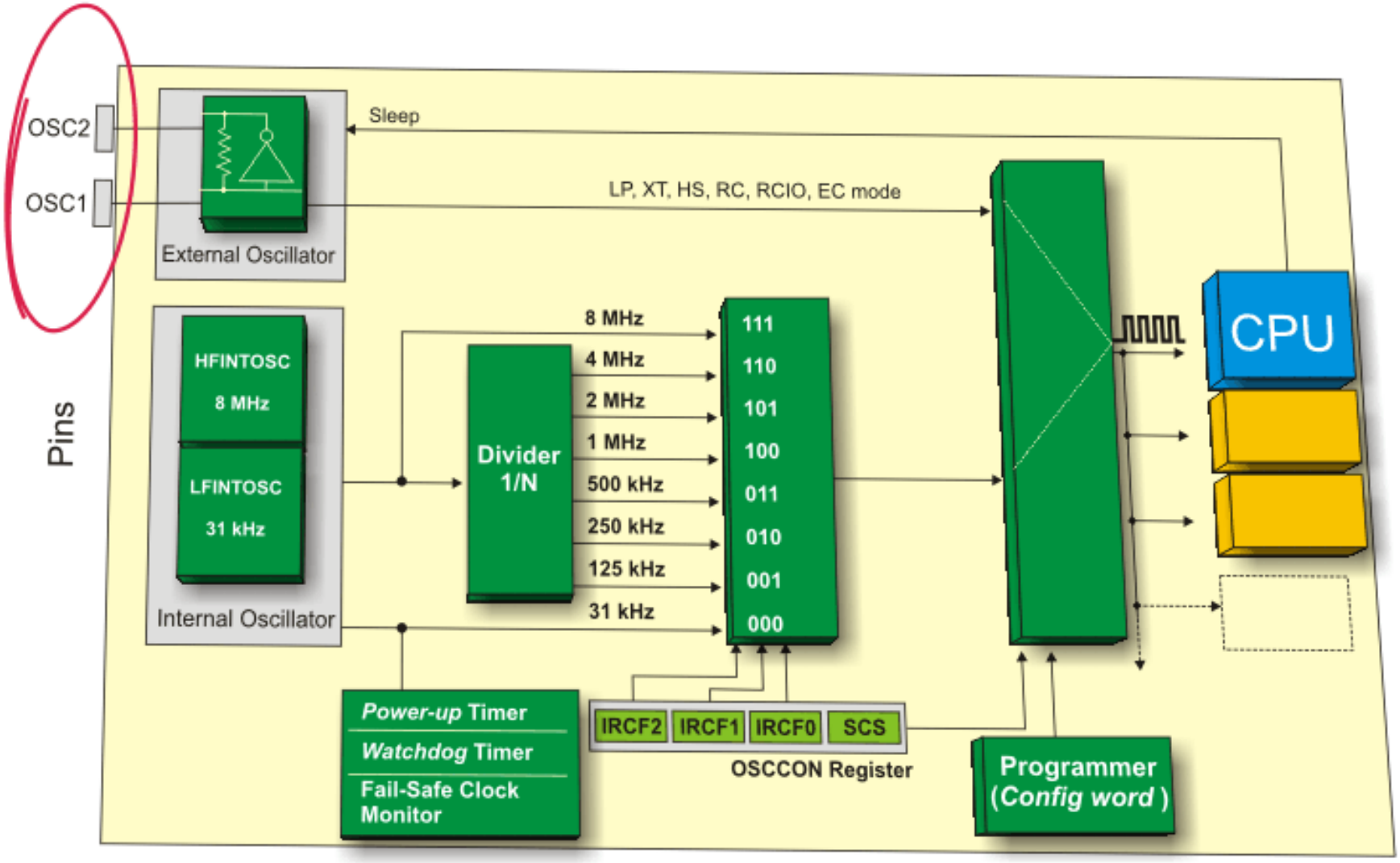


Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

Cadencement du microcontrôleur

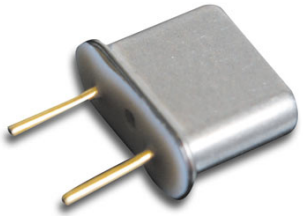
Schéma du pic 16F877



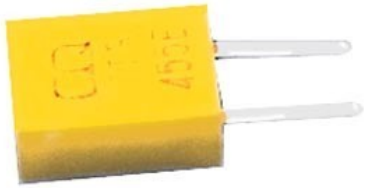
Cadencement du microcontrôleur

Les cadencements possibles (1)

- Quartz
  - Très bonne Stabilité (10 ppm)

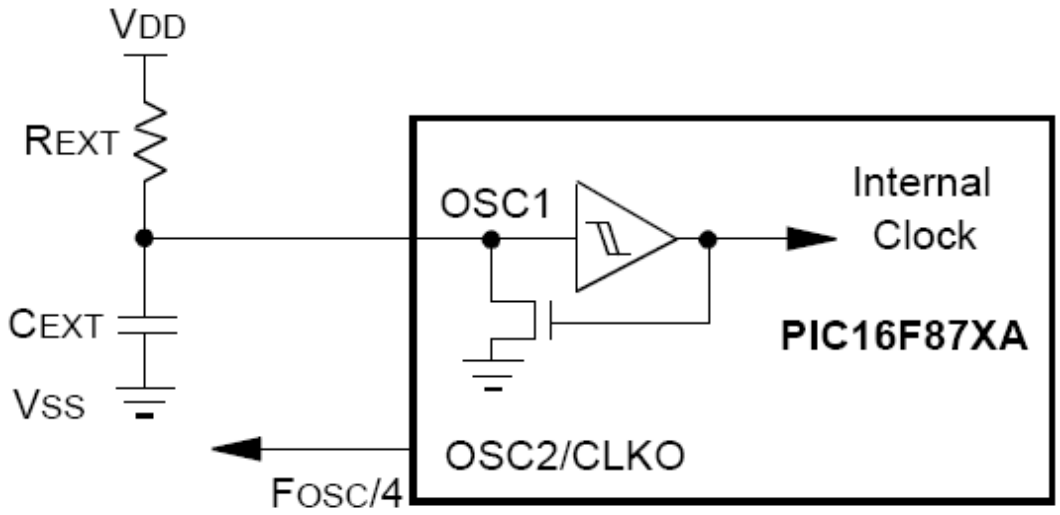


- Résonateur céramique
  - Stabilité (0.5%)
  - Moins couteux que le quartz



Cadencement du microcontrôleur
Les cadencements possibles (2)

- Externe
  - Permet de synchroniser plusieurs éléments du microsystème
- RC
  - Très peu stable mais très faible cout
  - Pas possible pour tous les microprocesseurs



Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

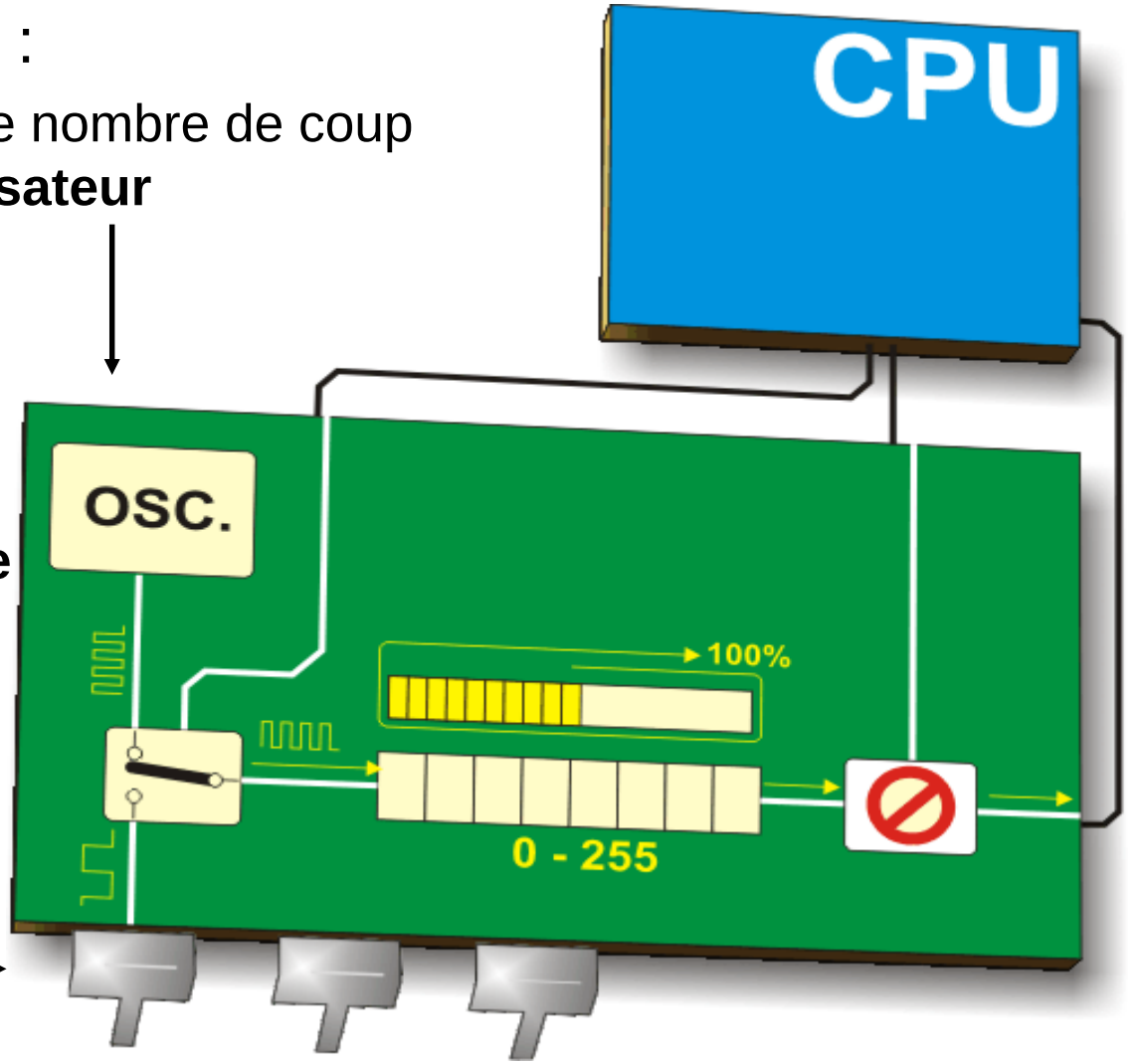
Les timers

Mode compteur ou temporisateur

- Un timer est le nom courant de compteur / temporisateur . Il sert à :

- Mesurer du temps (compter le nombre de coup d'horloge) > **Mode temporisateur**

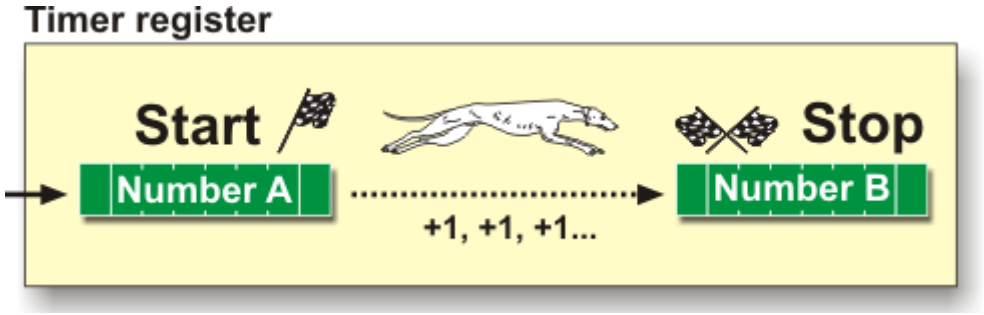
- Compter le nombre d'évènement sur une broche (exemple : Nombre d'appuis sur un bouton poussoir > **Mode compteur**



Les timers

Mode compteur ou temporisateur

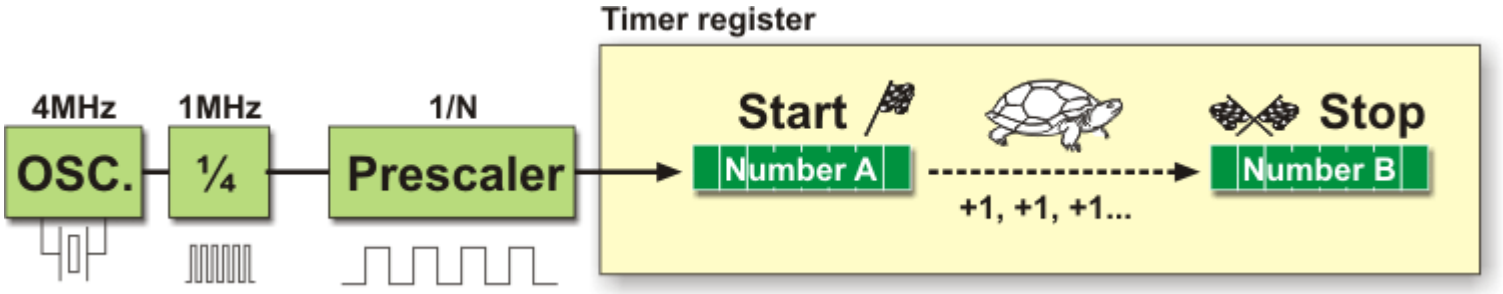
- En pratique, on visualise la valeur de départ, puis la valeur d'arrivée. La valeur de comptage est la différence des deux valeurs.



Les timers

Utilisation d'un prescaler

- Un prescaler permet de diviser la fréquence de comptage.



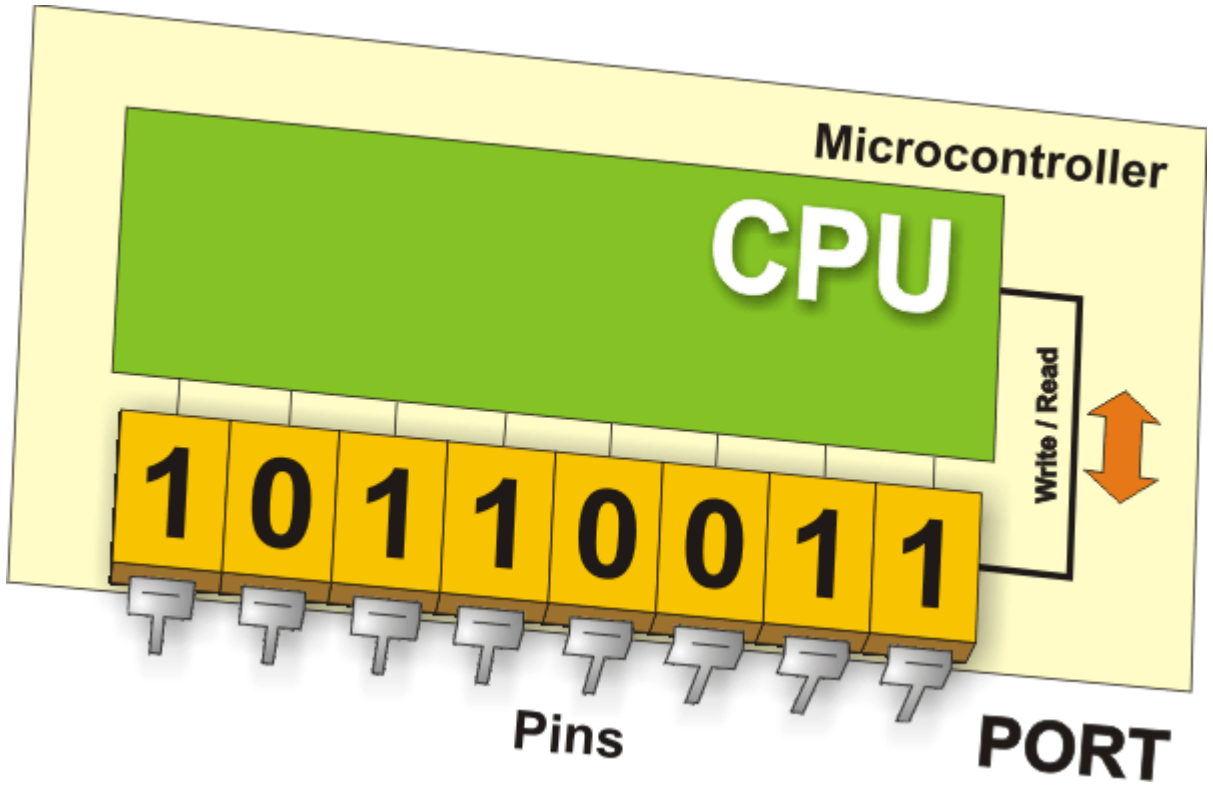
Quelle est le temps mesuré dans cette application en fonction de A et B?

Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

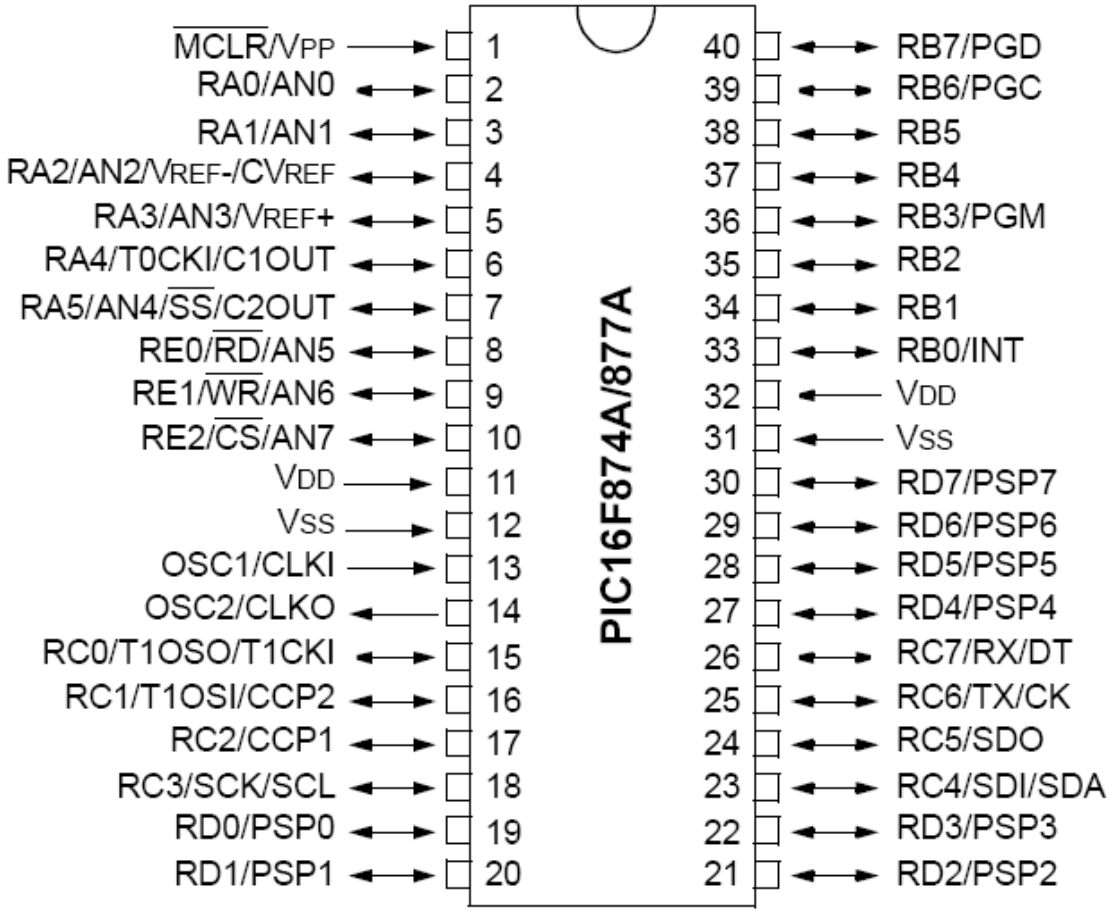
Les ports d’entrée / Sortie

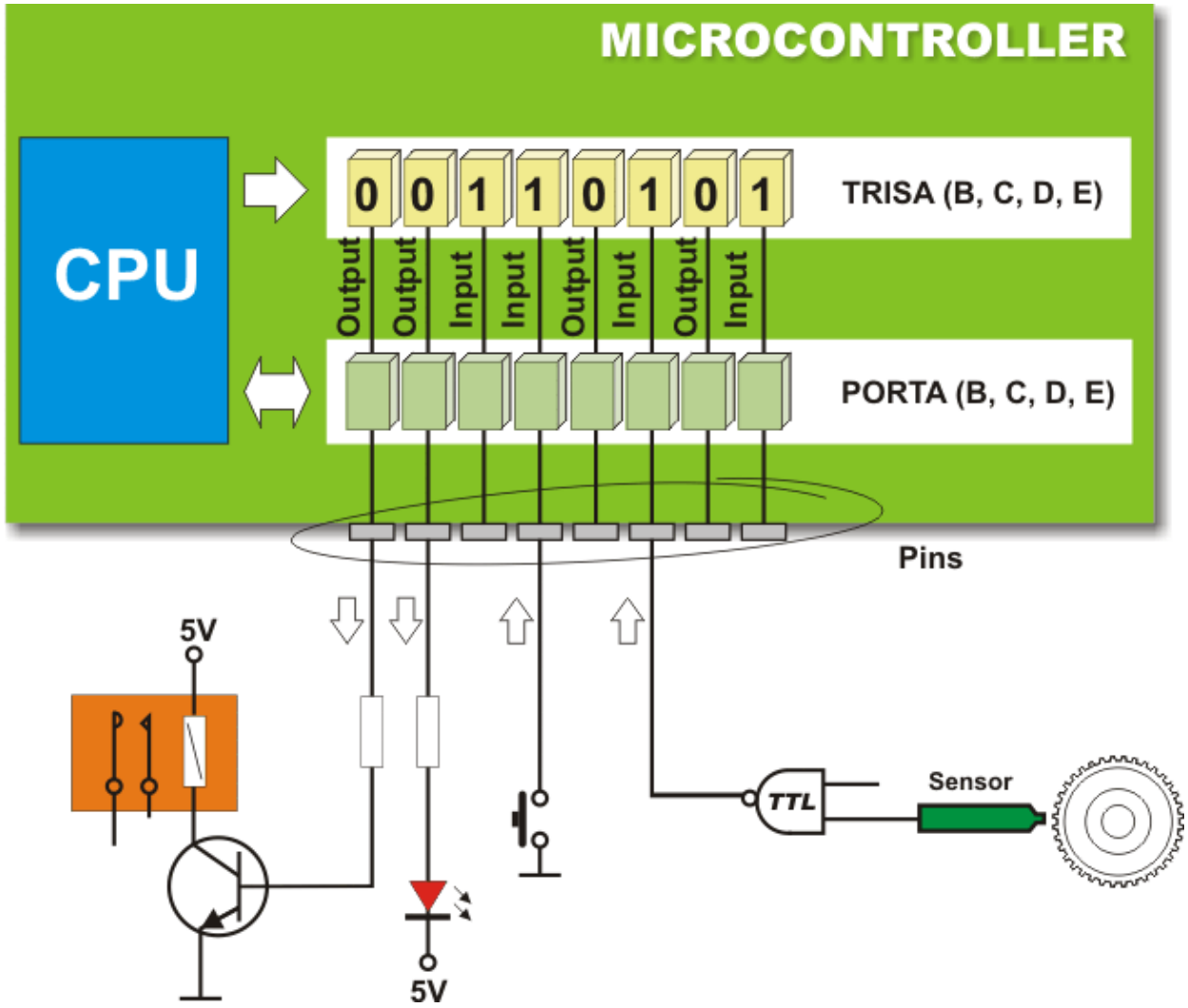
Ce sont des ports parallèles. Ils permettent de recueillir des informations ou de piloter des modules sur l’environnement extérieur. Ils sont souvent bidirectionnels (configurable en entrée ou sortie).



Les ports d'entrée / Sortie

- Quelles sont les ports d'E/S de ce microcontrôleur?





	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	Features
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	Features
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

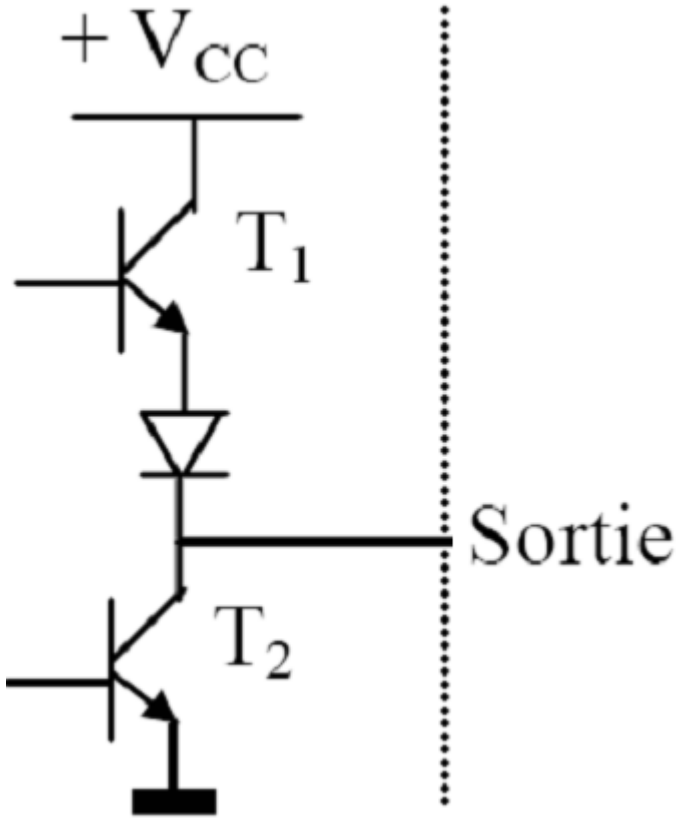
Les ports d’entrée / Sortie

Les sorties logiques

- Sortie trois états
- Sortie collecteur ouvert

Les ports d'entrée / Sortie

1. Sortie trois états

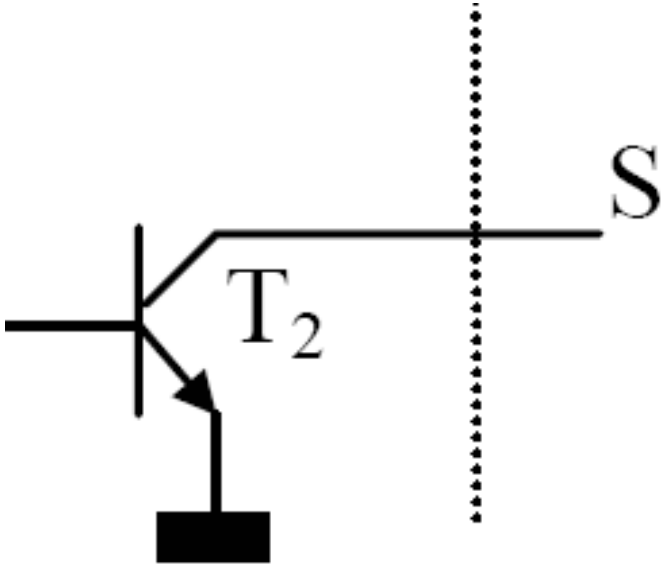


Fonctionnement		
T1	**T2**	**Sortie**
Bloqué	Bloqué	Etat haute impédance
Bloqué	Saturé	"0"
Saturé	Bloqué	"1"
Saturé	Saturé	non utilisé

- Des sorties trois états peuvent être reliées entres elles mais il faut bien veiller à ce que une seule impose un niveau (haut ou bas) et que les autres sorties soit en haute impédance.

Les ports d'entrée / Sortie

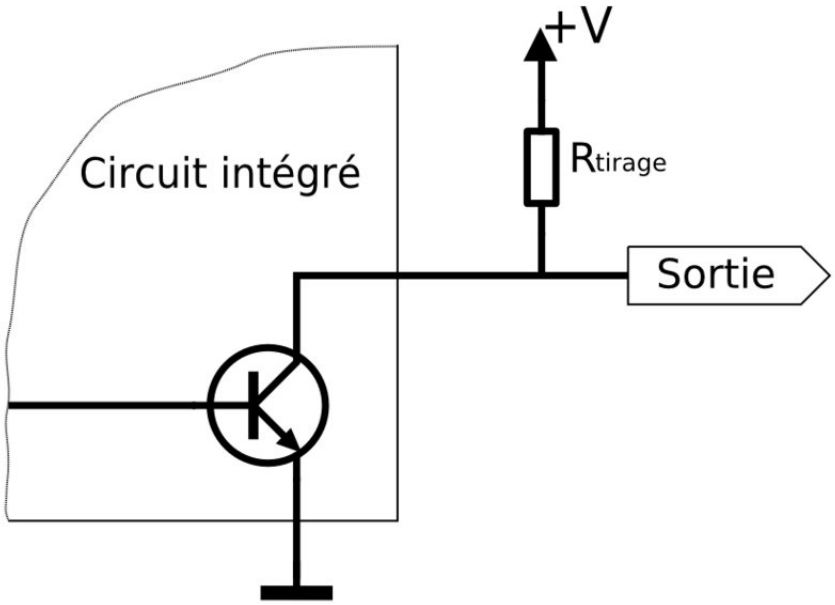
2. Sortie collecteur ouvert (1)



Fonctionnement	
T2	**S**
Saturé	"0"
Bloqué	Dépend du montage

Les ports d'entrée / Sortie

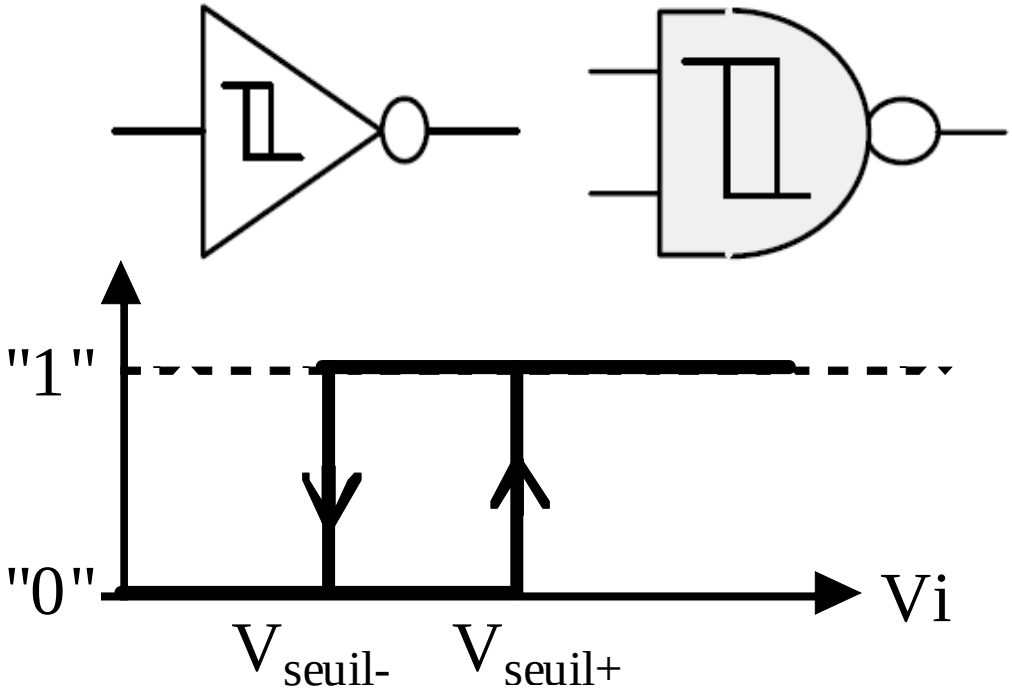
2. Sortie collecteur ouvert (2)



- Pour générer le niveau "1", une résistance extérieure est nécessaire (résistance de tirage // pull-up). Plusieurs sorties "collecteur ouvert" peuvent être reliées entre elles, cela réalise un "ET logique"
- Une sortie « collecteur ouvert » peut commander une charge sous une tension différente de la tension d'alimentation.

Les ports d'entrée / Sortie

Entrée trigger de Schmitt

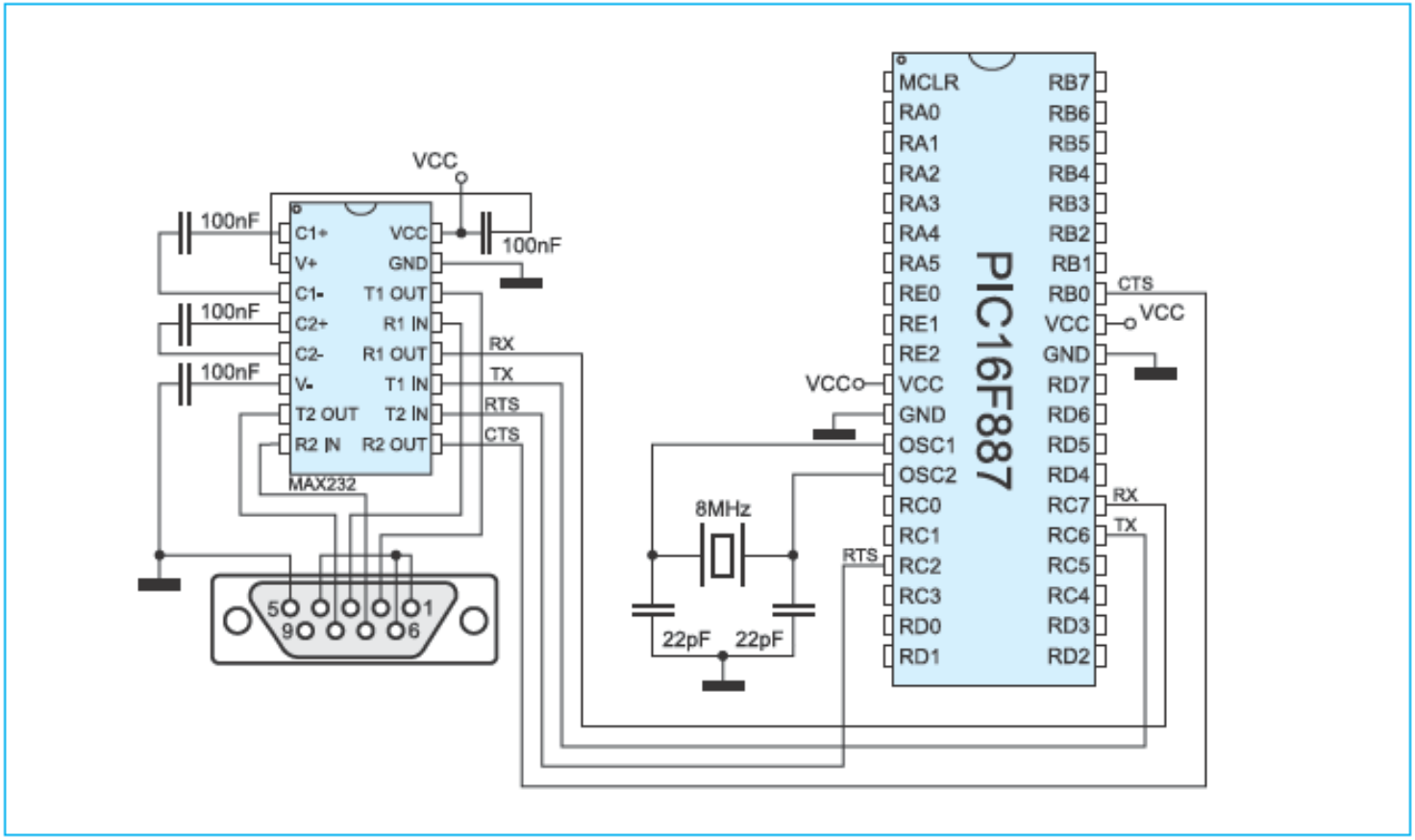


Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

La liaison série

- La liaison série USART (*Universal Synchronous Asynchronous Receiver Transmitter*) est le mode le plus rependu pour communiquer (et aussi le plus vieux).



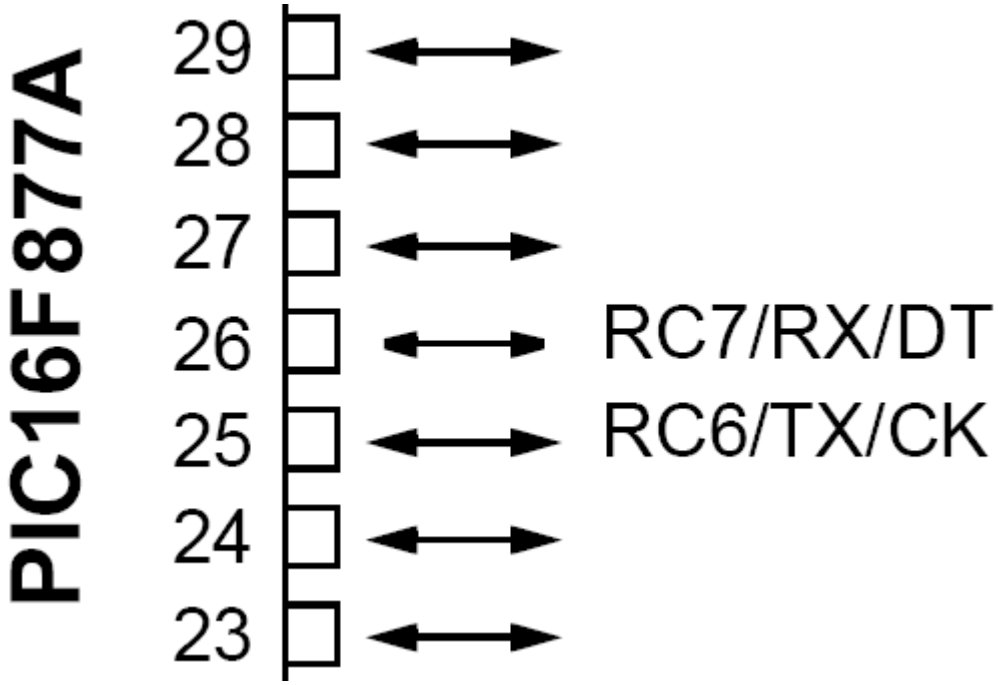
La liaison série
Connexions (1)

- Une liaison série permet de transmettre des données sur un nombre limité de fils. Cette liaison peut être Half duplex (liaison synchrone) ou full duplex (liaison asynchrone).

- Sur le microcontrôleur PIC16F877A, la liaison série asynchrone peut être mise en œuvre à l'aide des pins RxD et TxD. Elle est réalisée par un USART :
  - RxD, signal de réception de l'USART
  - TxD, signal de transmission de l'USART

La liaison série

Connexions (2)

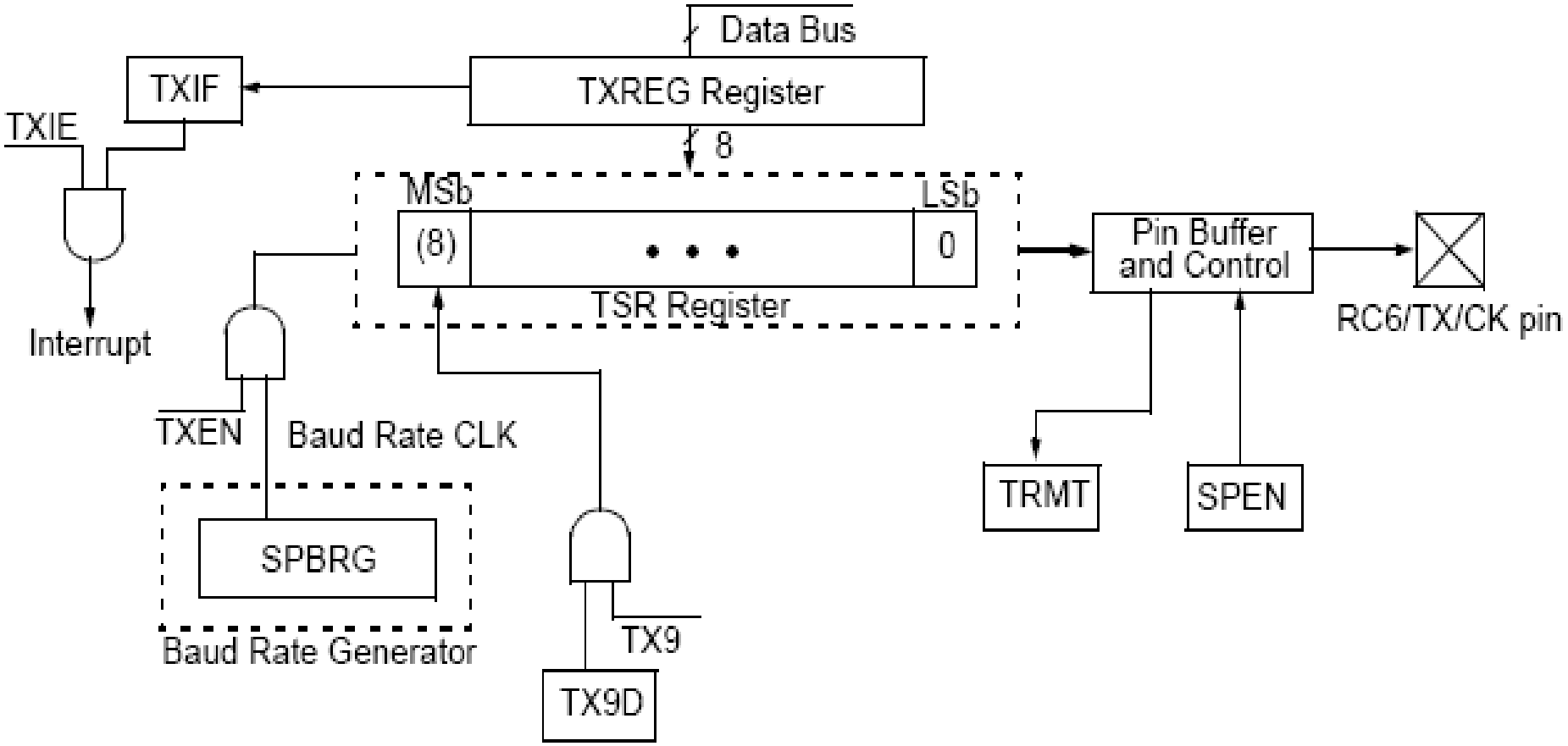


1ère étape : Configuration de la transmission

2ième étape : Envoyer et recevoir des données

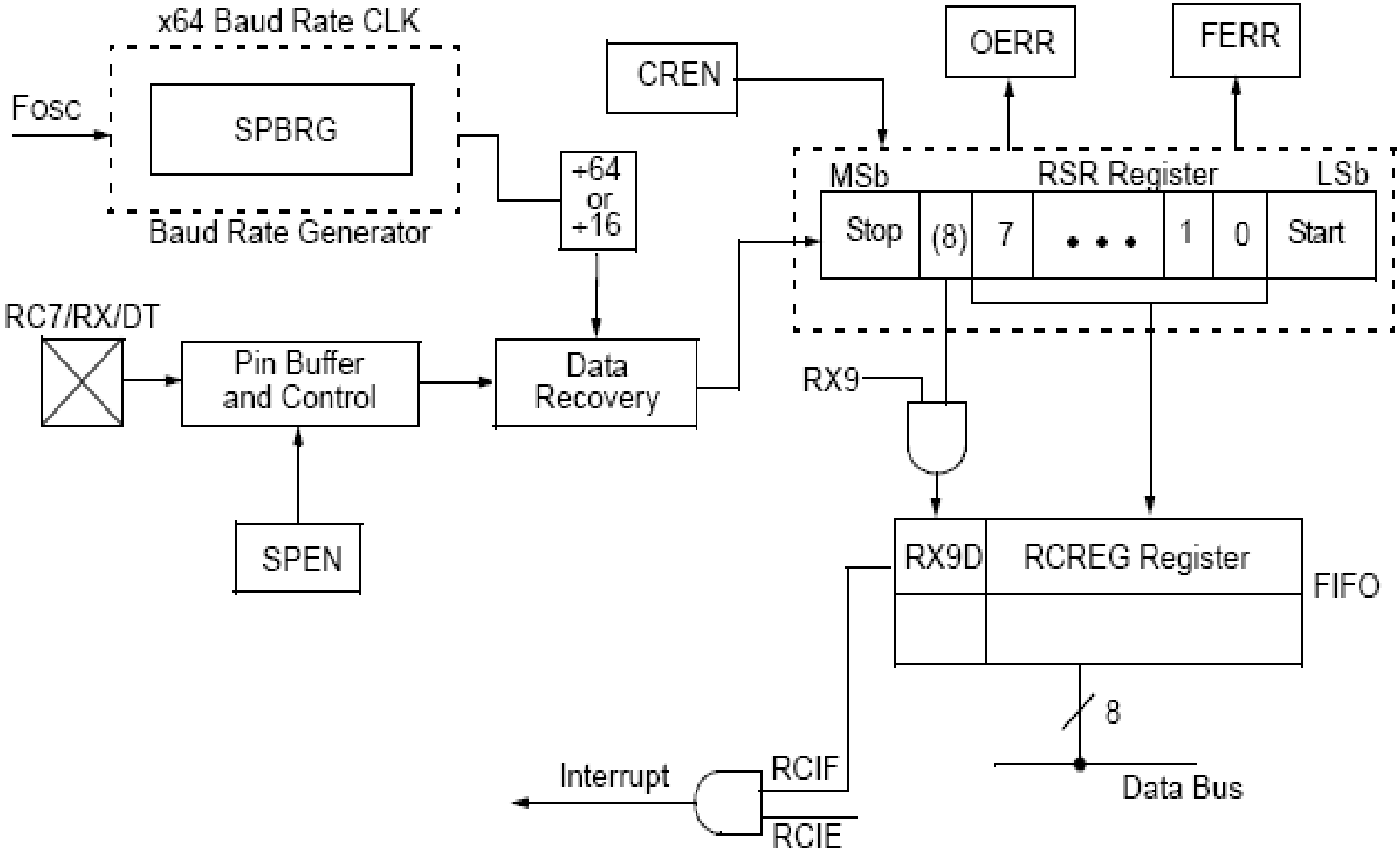
La liaison série

USART Transmit bloc register



La liaison série

USART receive bloc register



Chapitre 2 : Les microcontrôleurs

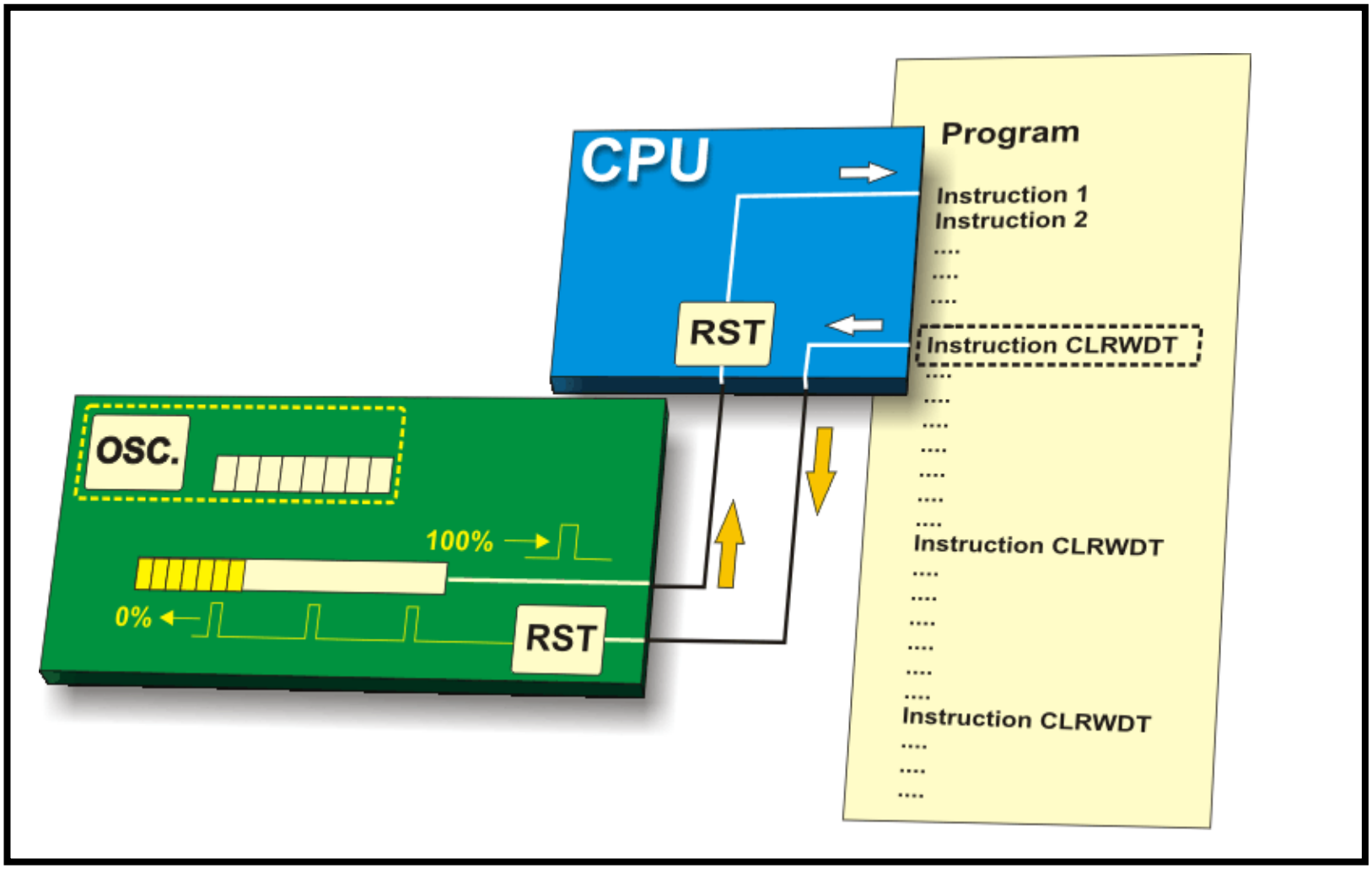
- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

Le Watchdog (1)

Le chien de garde (watchdog) est un dispositif matériel et logiciel qui permet de se prémunir contre les plantages accidentels. L'idée est de provoquer un RESET du CPU afin de relancer l'application. (Les données sont bien sur perdues). Le plantage est défini lorsque le programme n'est pas venu à temps faire signe au watchdog.

Le Watchdog (2)

MICROCONTROLEUR



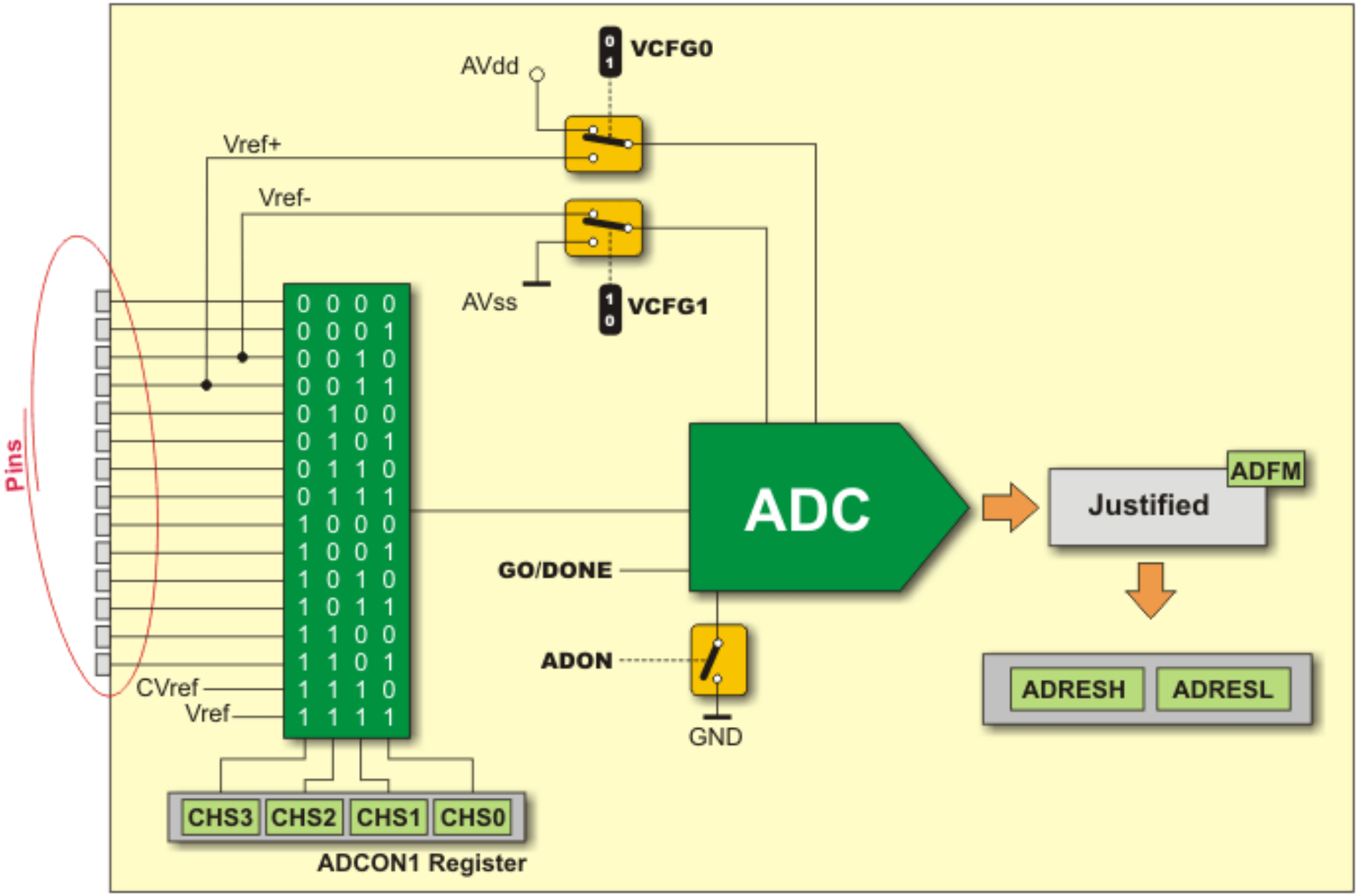
Chapitre 2 : Les microcontrôleurs

- 2.1 Définition d'un microcontrôleur
- 2.2 Cadencement du microcontrôleur
- 2.3 Les timers
- 2.4 Les ports d'entrée/sortie
- 2.5 La liaison série
- 2.6 Le watchdog
- 2.7 Le CAN

Le CAN

- CAN : Dans les microcontrôleurs, les voix de conversion analogique/numérique sont souvent multiplexées. Ceci signifie que la fréquence maximale de conversion analogique numérique est divisée par le nombre de voies utilisées.

- Très souvent, il faut configuré les entrées de conversion en « entrée analogique » car celles-ci peuvent aussi être utilisée en entrée numérique.


AVdd
VCFG0
Vref+
Vref-
AVss
VCFG1
Pins
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
CVref
Vref
ADC
GO/DONE
ADON
GND
ADFM
Justified
ADRESH
ADRESL
CHS3
CHS2
CHS1
CHS0
ADCON1 Register

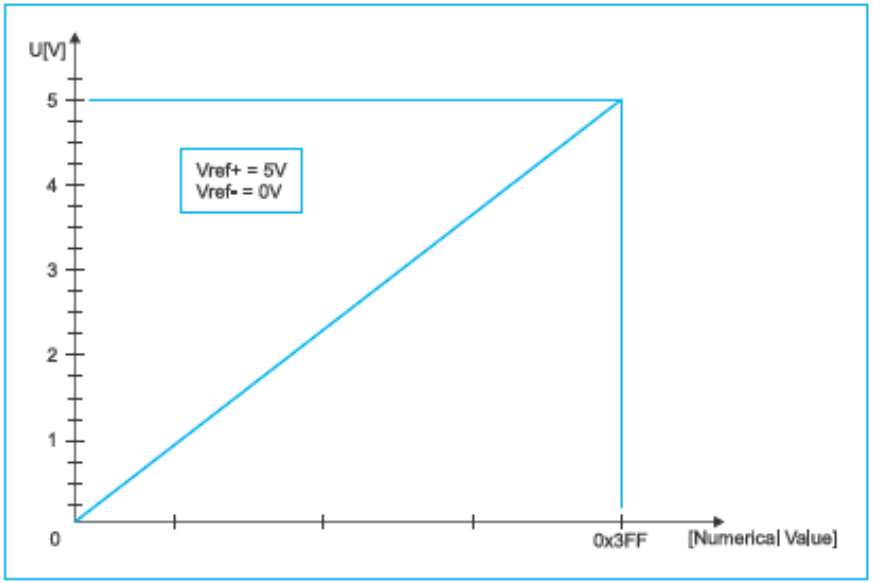
Le CAN

D’après la datasheet :

- Quel est le nombre de bits du convertisseur AN.
- Dans quels registres est stocké le résultat?
- Comment fait-on pour justifier le résultat à droite ou à gauche.
- Combien possède t on de voies multiplexés?
- A quoi correspond Vref+ et Vref - ?
- Comment configure t on une entrée en analogique?

Le CAN

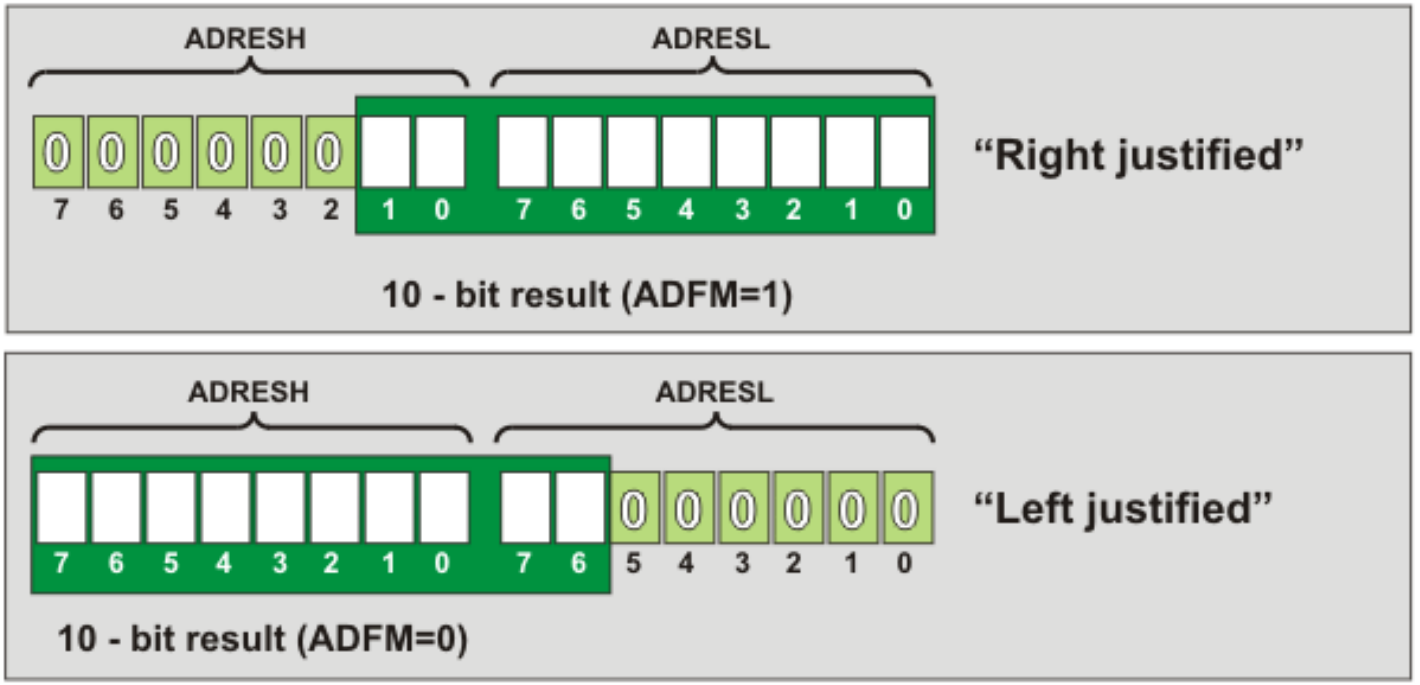
- Sur l'application ci-dessous, quelle est la résolution maximale que nous pourrons avoir?



- Donner l'expression de la résolution en fonction de Vref+, Vref- et du nombre de bits du convertisseur.
- Exprimer la valeur analogique en fonction de Vref+, Vref- et de la valeur numérique donnée par le convertisseur.

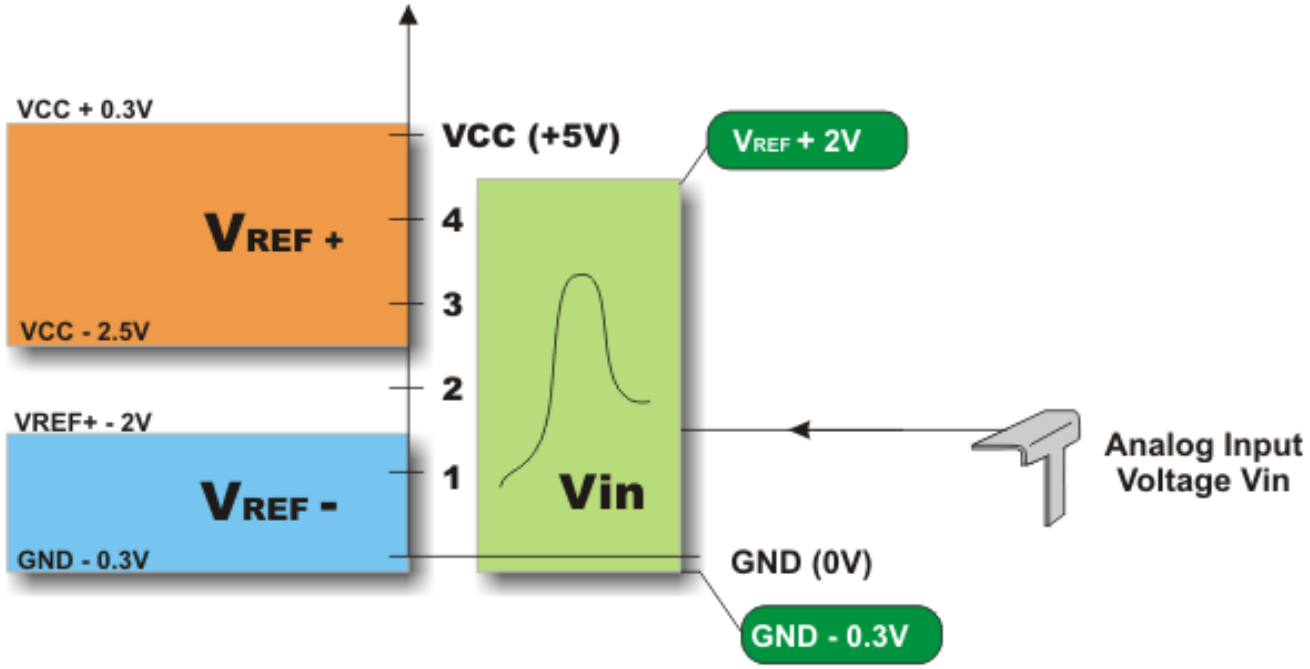
Le CAN

- Donner le code C permettant d'obtenir un entier représentatif de la valeur de conversion en fonction des valeurs de ADRESH et ADRESL.



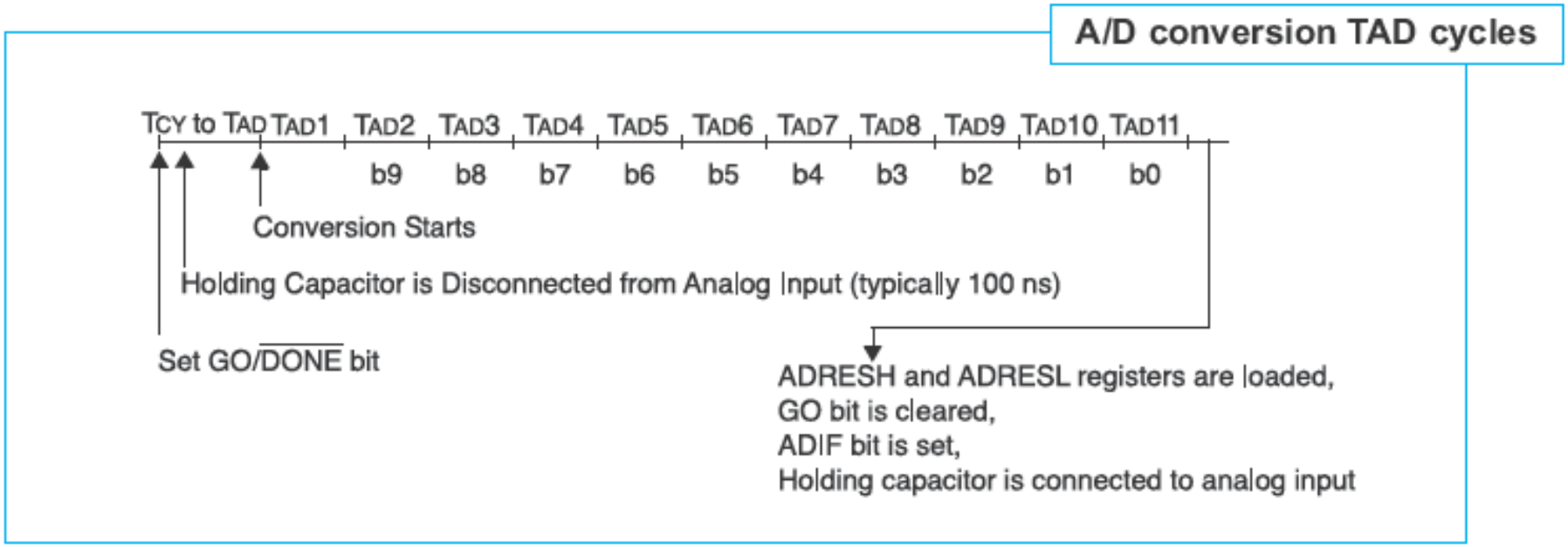
Le CAN

- Dans l'application ci-dessous, donner :
  - La résolution
  - L'expression de Vin en fonction de la valeur numérique du convertisseur.



Le CAN

- Vitesse maximale du convertisseur



Gamme des microcontrôleurs

- Exemple chez microchip

http://www.microchip.com/

- Exemple chez ATMEL

http://www.atmel.com/

- Exemple chez ARM

http://www.arm.com/

Etc …

Chapitre 3 : La programmation

- 3.1 Les interruptions
- 3.2 Le logiciel

Les interruptions
Problématique & définition

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

Deux modes sont possibles :

- Une méthode par scrutation (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
- Une méthode par interruption permet au périphérique lui-même de faire signe au processeur de sa présence.

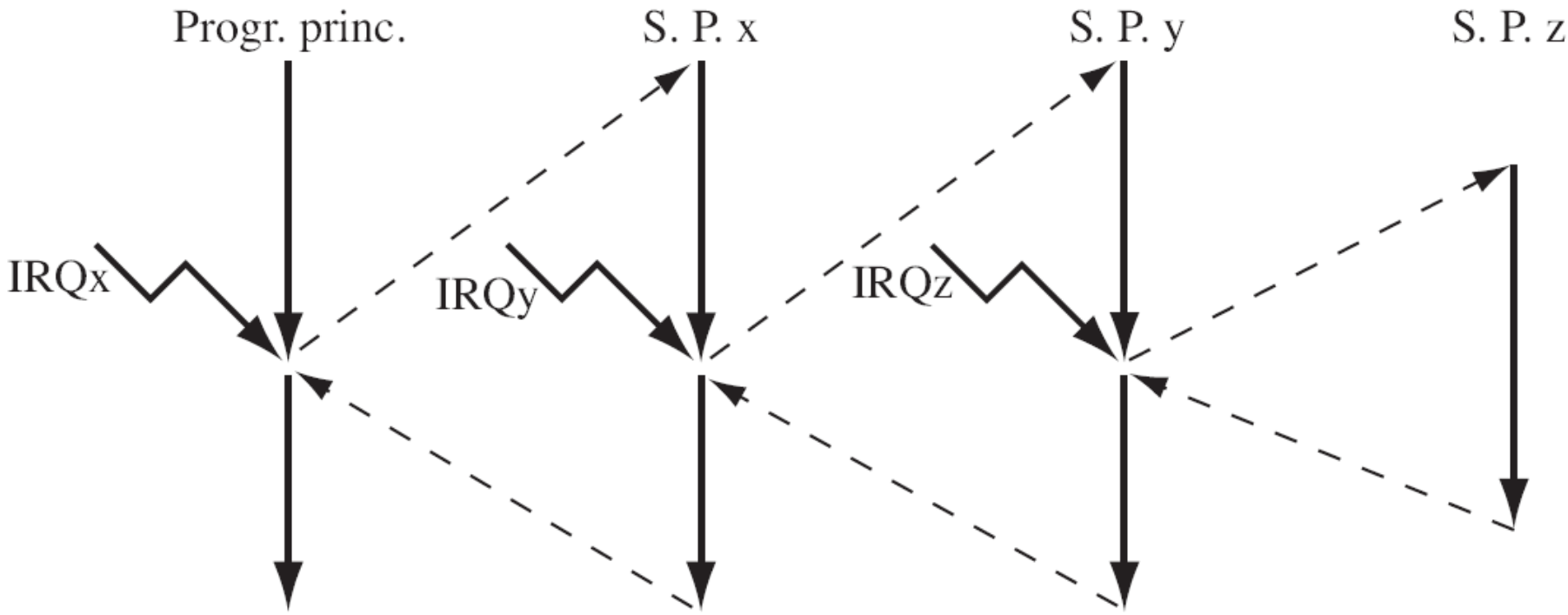
Les interruptions
Scrutation Vs interruption

- Scrutation (polling)
  - 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

Les interruptions
Schéma

- 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).



Les interruptions
Types d'interruption

- Interruption masquable
  - Un masque d'interruption est un mot binaire de configuration du microprocesseur qui permet de choisir **(démasquer)** quels modules pourront interrompre le processeur parmi les interruptions disponibles.

- Interruption non masquable
  - Elles s 'exécutent quoi qu'il arrive, souvent avec une priorité élevé (ex : Reset)

Les interruptions
Configuration

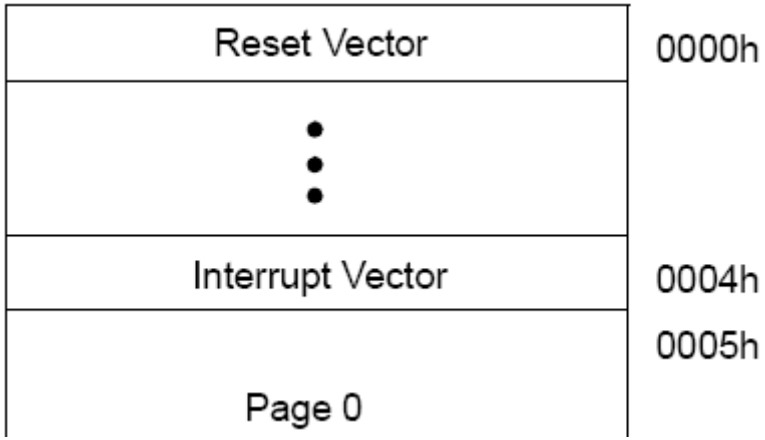
- Un système peut accepter plusieurs sources d'interruption. Chacune est configurable par registre (registre d'interruption).

- Méthode de configuration des interruptions
  - Sélectionner les interruptions qui nous intéressent
  - Valider les interruptions de façon globale
  - Ecrire le/les sous programme d'interruption
  - Définir les priorités entres interruptions

Les interruptions
Configuration

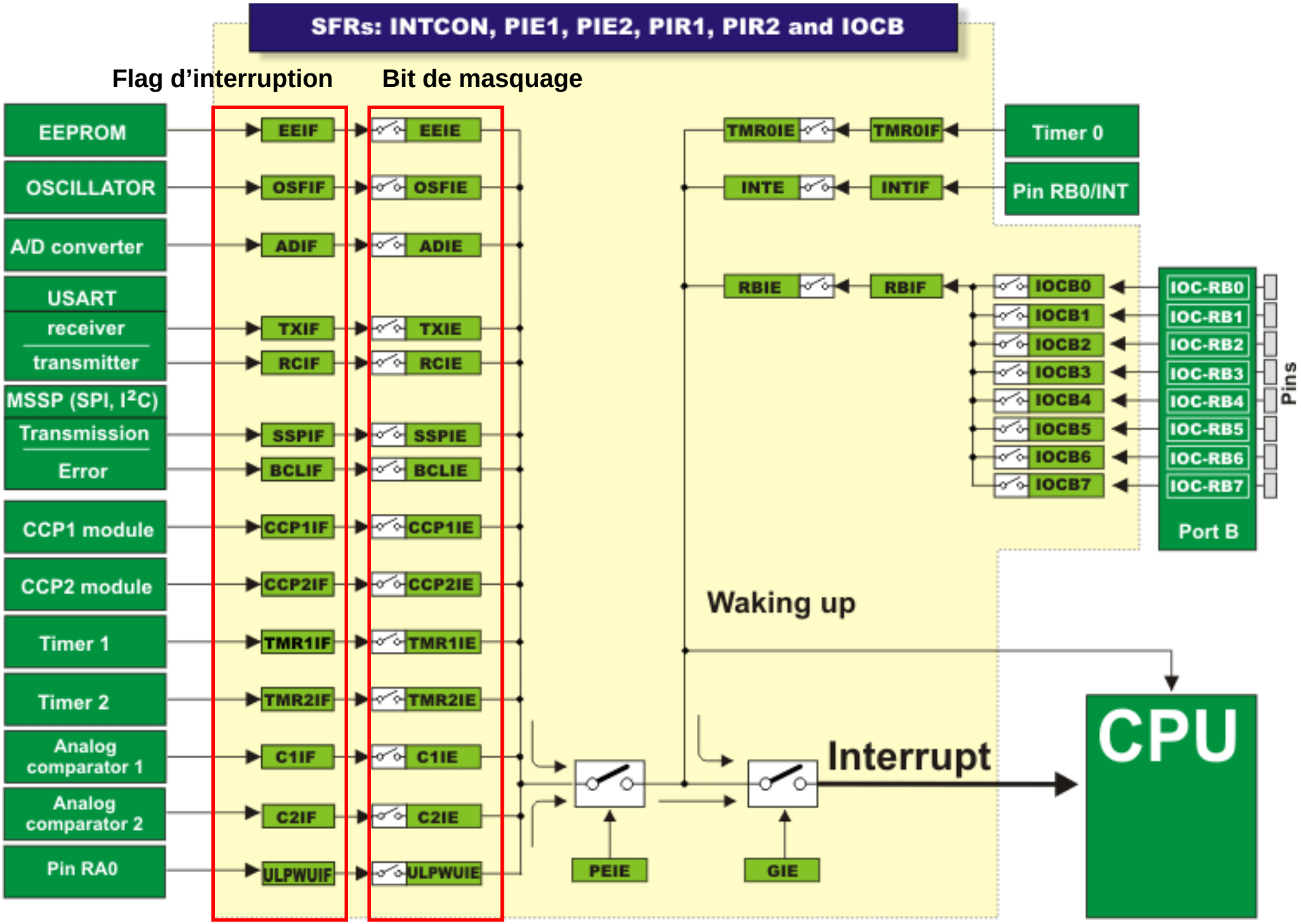
- Dans le sous programme d'interruption
  - Sauvegarder le contexte (fait automatique en langage C)
  - Définir la source d'interruption (si le sous programme est commun entres plusieurs sources d'interruption)
  - Réinitialiser les flags d'interruption
  - Ecrire le code relatif à l'application
  - Restituer le contexte (fait automatique en langage C)

Cas du PIC 16F877 (microchip)



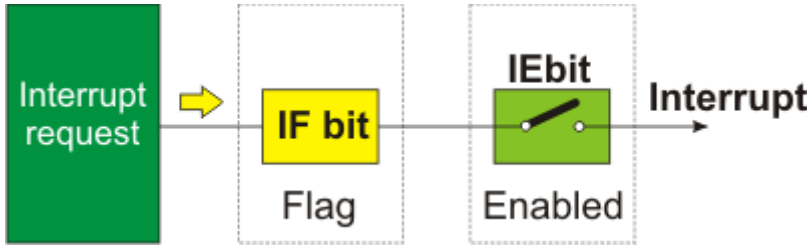
Cas du 80C51 (intel)

Interrupt Source	Interrupt Request Bits	Cleared by Hardware	Vector Address
$\overline{\text{INT0}}$	IE0	No (level) Yes (trans.)	0003H
TIMER 0	TF0	Yes	000BH
$\overline{\text{INT1}}$	IE1	No (level) Yes (trans.)	0013H
TIMER 1	TF1	Yes	001BH
SERIAL PORT	RI, TI	No	0023H
TIMER 2	TF2, EXF2	No	002BH


SFRs: INTCON, PIE1, PIE2, PIR1, PIR2 and IOCB
Flag d'interruption
Bit de masquage
EEPROM
OSCILLATOR
A/D converter
USART
receiver
transmitter
MSSP (SPI, I2C)
Transmission
Error
CCP1 module
CCP2 module
Timer 1
Timer 2
Analog comparator 1
Analog comparator 2
Pin RA0
EEIF
OSFIF
ADIF
TXIF
RCIF
SSPIF
BCLIF
CCP1IF
CCP2IF
TMR1IF
TMR2IF
C1IF
C2IF
ULPWUIF
EEIE
OSFIE
ADIE
TXIE
RCIE
SSPIE
BCLIE
CCP1IE
CCP2IE
TMR1IE
TMR2IE
C1IE
C2IE
ULPWUIE
TMR0IE
INTE
RBIE
TMR0IF
INTIF
RBIF
Timer 0
Pin RB0/INT
IOCB0
IOCB1
IOCB2
IOCB3
IOCB4
IOCB5
IOCB6
IOCB7
IOC-RB0
IOC-RB1
IOC-RB2
IOC-RB3
IOC-RB4
IOC-RB5
IOC-RB6
IOC-RB7
Pins
Port B
Waking up
PEIE
GIE
Interrupt
CPU

Les interruptions
Démasquage des interruptions

- Autorisation des interruptions
  - L'autorisation globale des interruptions



  - Démasquage des interruptions

INTCON	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)				
	GIE	PEIE	T0IE	INTE	RBIE				
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3				

PIE1		R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	Features
	-	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

PIE2	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)		R/W (0)	Features
	OSFIE	C2IE	C1IE	EEIE	BCLIE	ULPWUIE	-	CCP2IE	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

Les interruptions
Les flags d'interruption

- Visualisation des flags d'interruption

INTCON								
						R/W (0)	R/W (0)	R/W (x)
						T0IF	INTF	RBIF
						Bit 2	Bit 1	Bit 0

Features: Bit name

PIR1								
	R/W (0)	R (0)	R (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	
	-	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

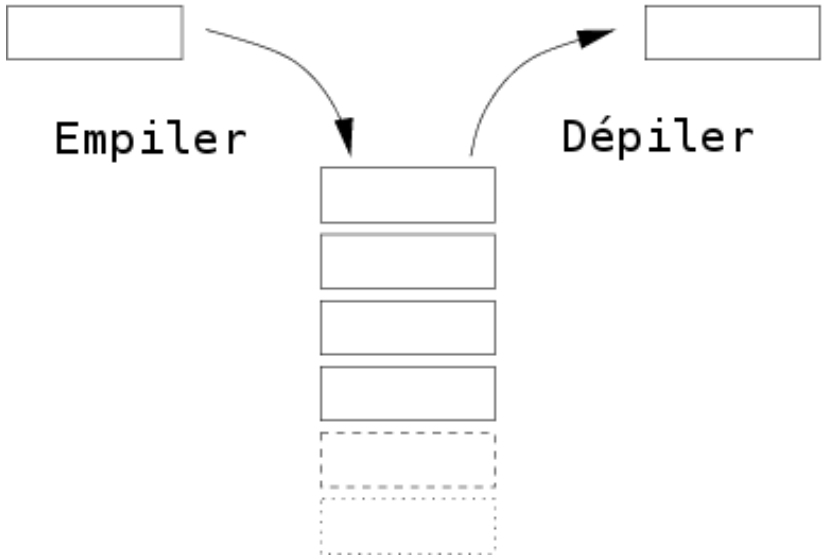
Features: Bit name

PIR2								
	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)		R/W (0)
	OSFIF	C2IF	C1IF	EEIF	BCLIF	ULPWUIF	-	CCP2IF
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Features: Bit name

Les interruptions
Le rôle de la pile

- La pile est une mémoire LIFO (Last In First Out) dans laquelle on stoke des variable temporaire (donnée ou adresse). Le haut de la pile est pointé par le registre SP (Stack Pointer).



Les interruptions
Rôle de la pile

- Elle va servir à :
  - **sauvegarder le contexte** l'environnement (adresse du programme et valeur des registres au moment de l'interruption).
  - **restituer le contexte** à la fin de l'interruption

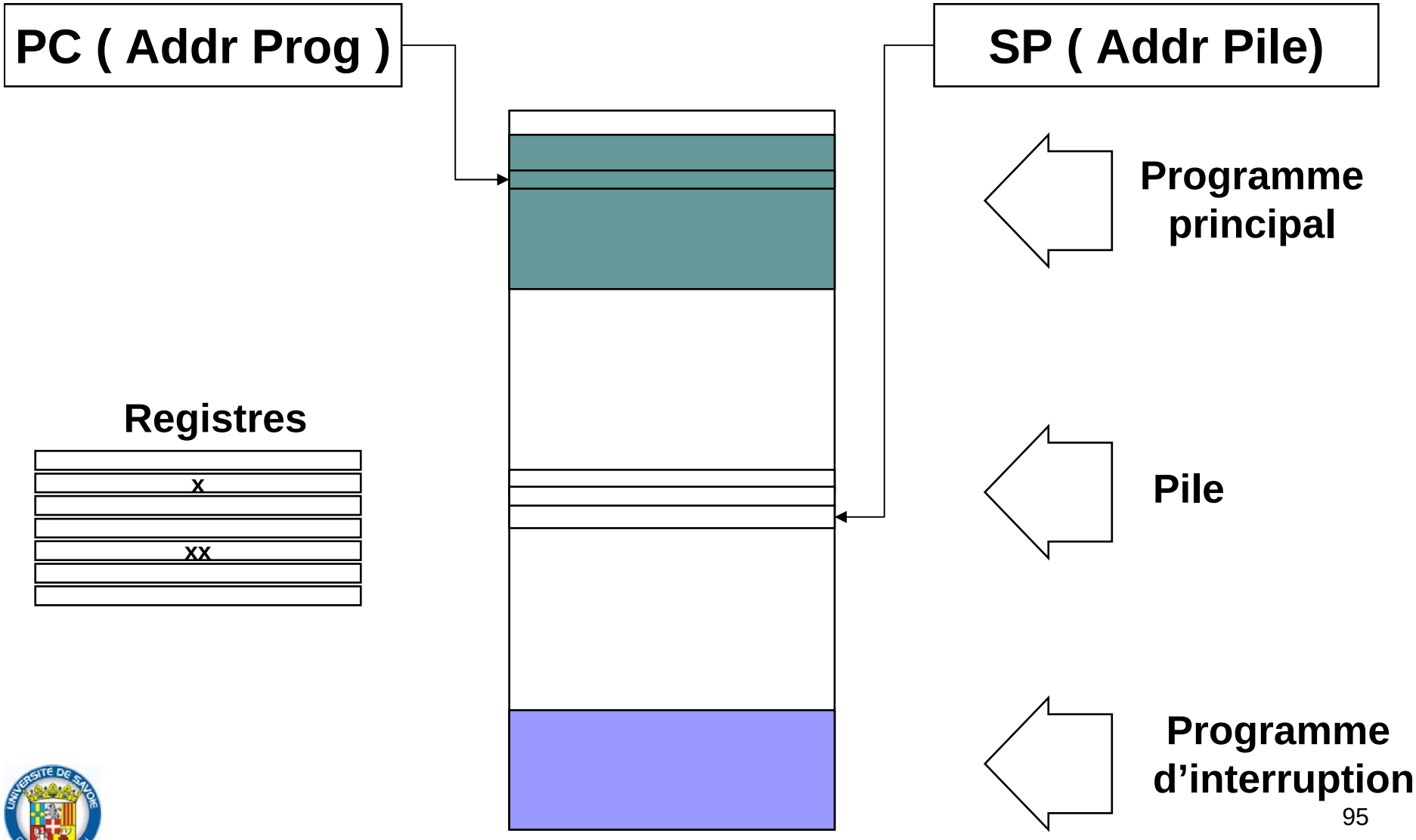
Note 1 : La sauvegarde et la restitution est faite implicitement en langage C.

Note 2 : Une fonction d'interruption est noté spécifiquement. Exemple du PIC qui ne possède qu'un seul vecteur d'interruption :

```
void interrupt() {
}
```

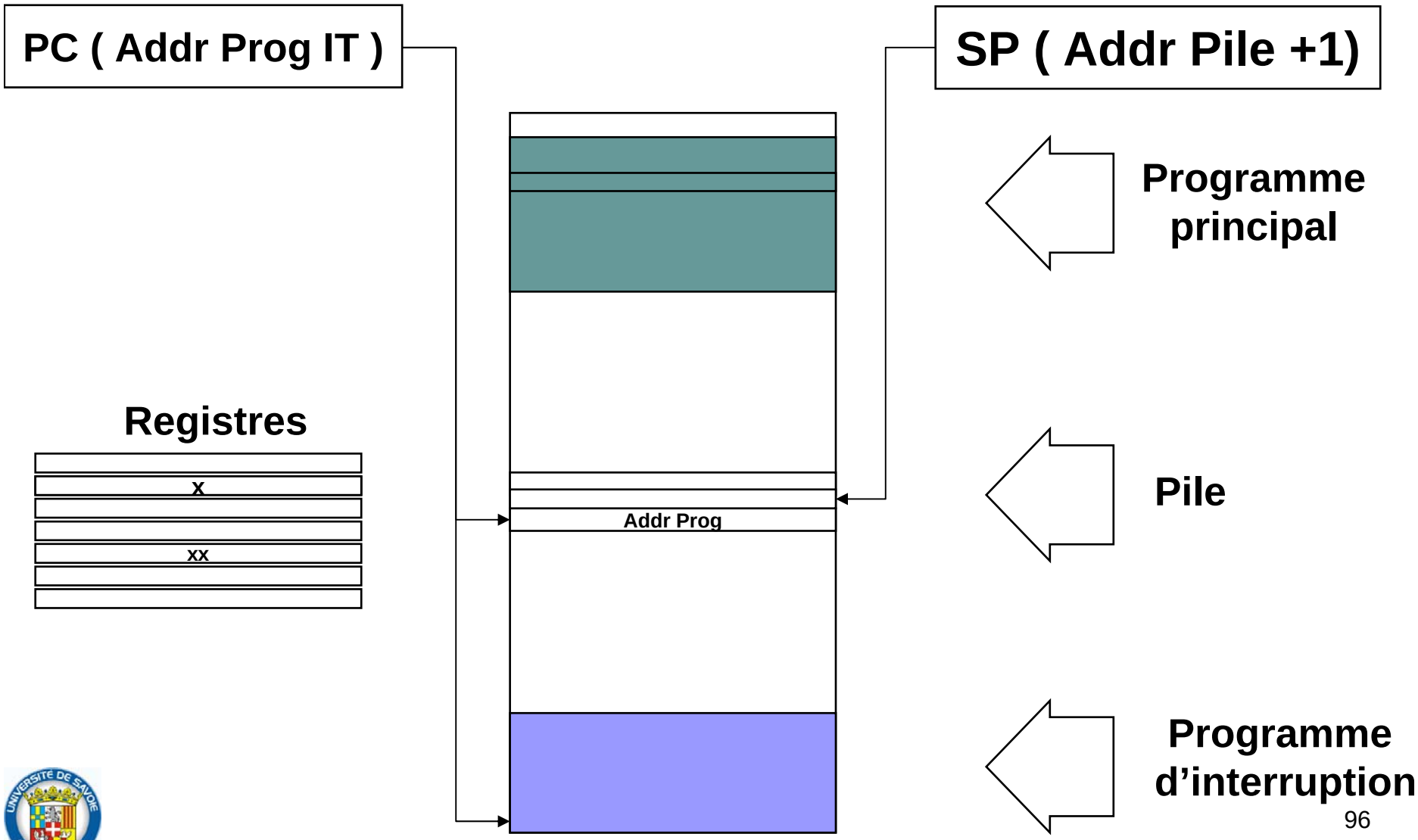
Les interruptions

Avant l'interruption



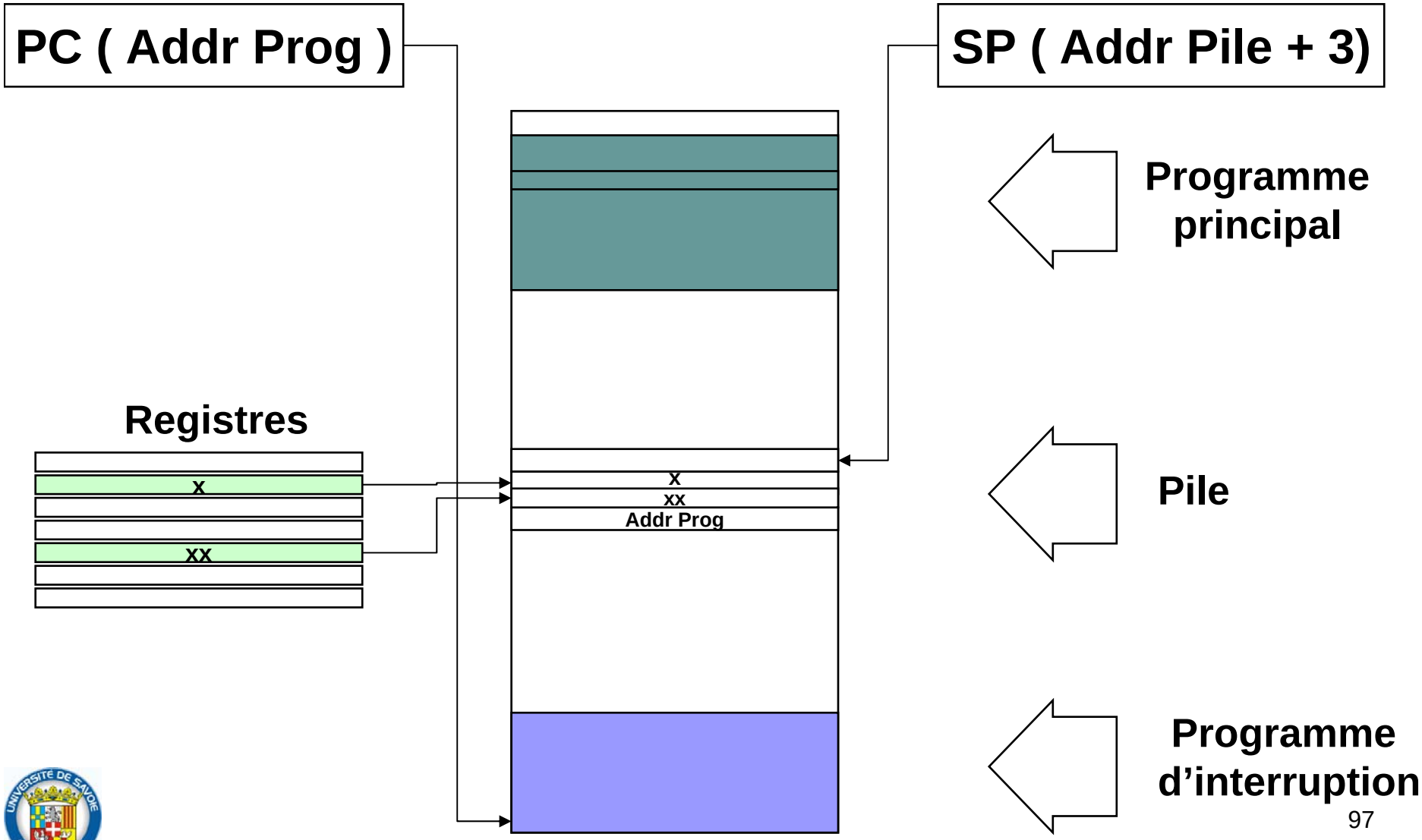
Les interruptions

Arrivée d'une interruption



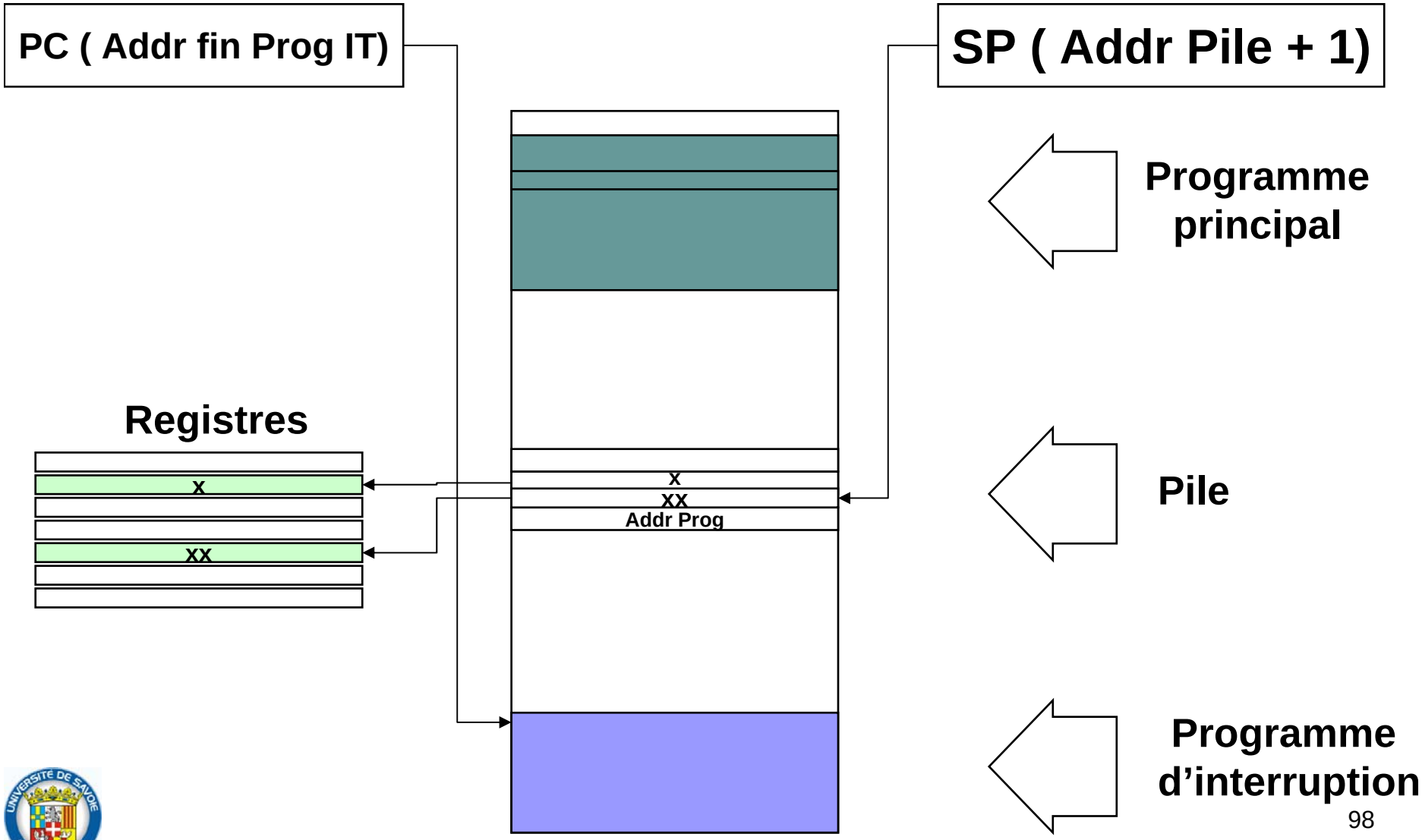
Les interruptions

Arrivée d'une interruption : Sauvegarde contexte



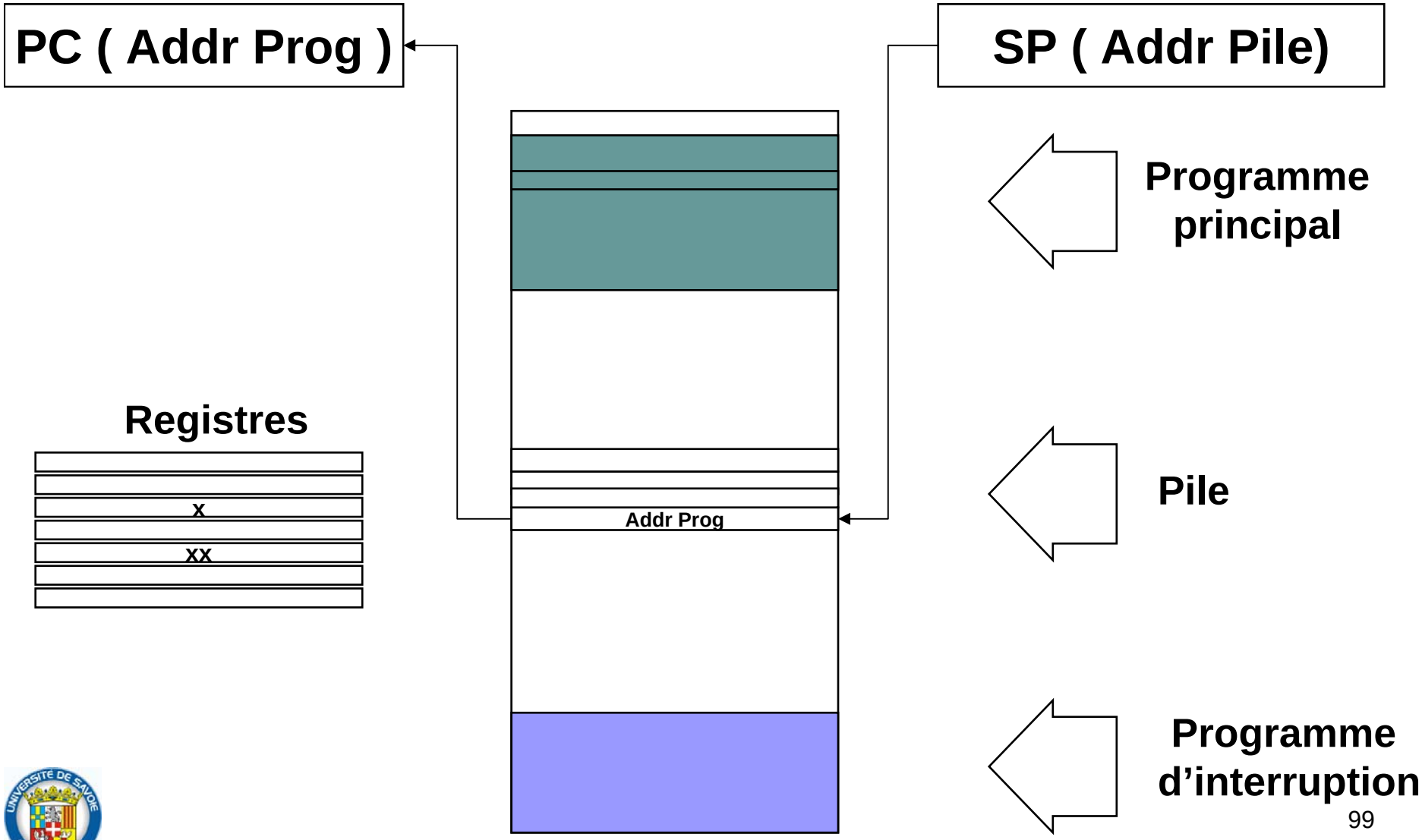
Les interruptions

Fin d'une interruption : Restitution contexte



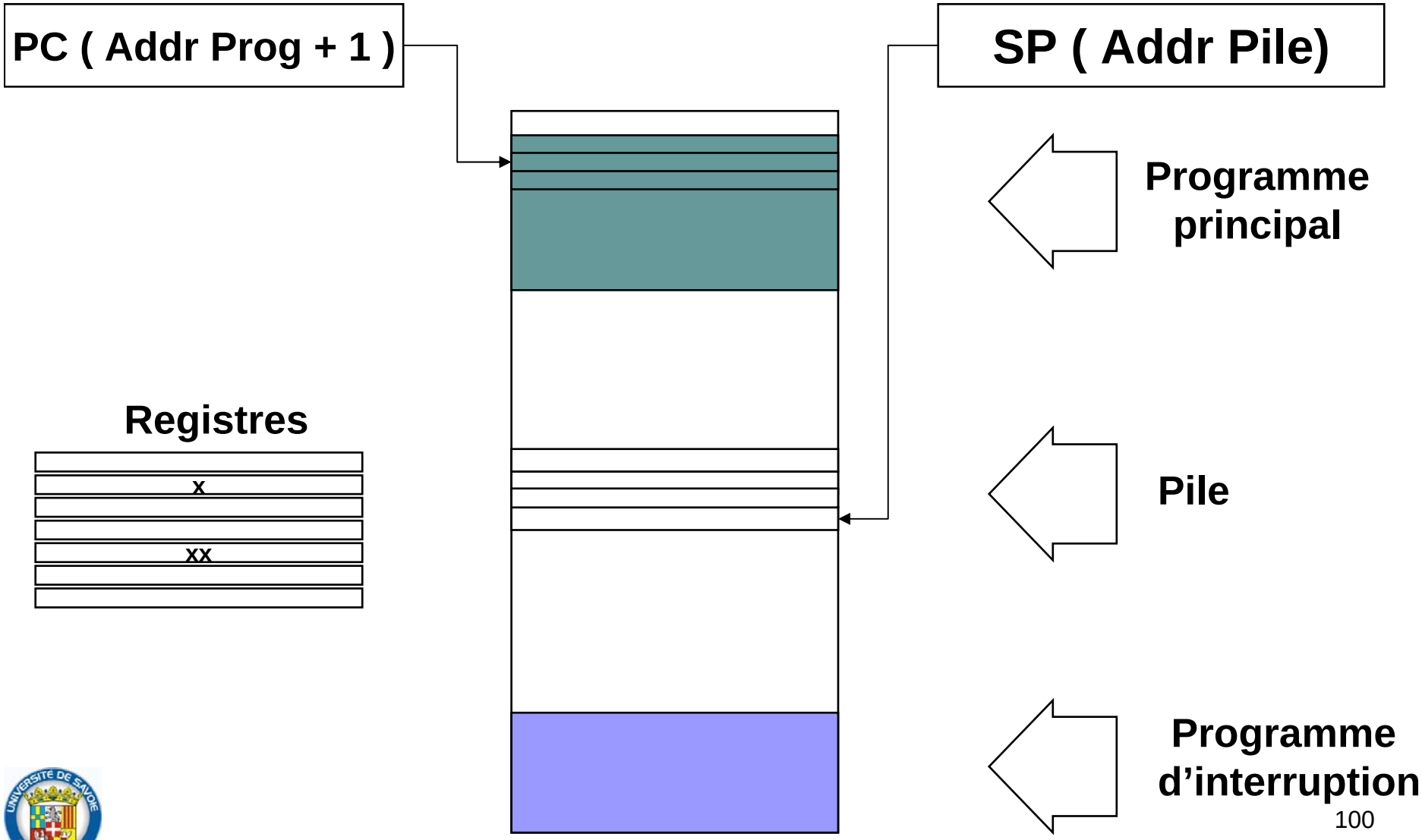
Les interruptions

Fin d'une interruption



Les interruptions

Retour au programme principal



Les interruptions
Exemple sur le PIC 16F877

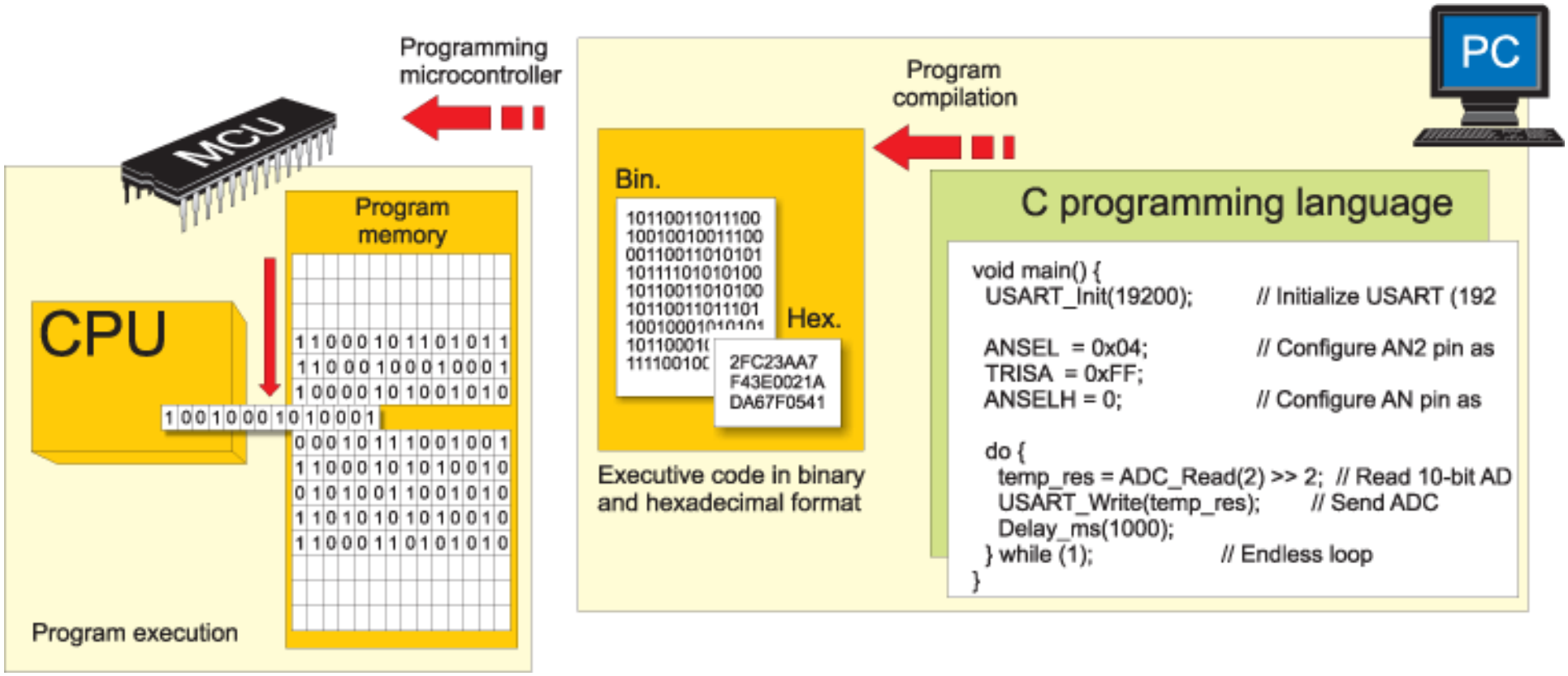
- Quelle interruption est concernée ici ?
- Quelles actions sont réalisées pendant le sous-programme d'interruption?

```
unsigned short cnt; // Define variable cnt

void interrupt() {
    cnt++ ;                    // Interrupt causes cnt to be incremented by 1
    PIR1.TMR1IF = 0;           // Reset bit TMR1IF
    TMR1H = 0x80;              // TMR1H and TMR1L timer registers are returned
    TMR1L = 0x00;              // their initial values
}
```

Chapitre 3 : La programmation

- 3.1 Les interruptions
- 3.2 Le logiciel

Le logiciel
La chaîne de compilation



```
void main() {

   TRISB = 0;                 // All port B pins are configured as
                              // outputs
   PORTB = 0b01010101;        // Logic state on port B pins
}
```

Program written in C

Remplacer les deux lignes du programme C par le code assembleur correspondant

Donner le code machine correspondant à chaque instruction trouvée

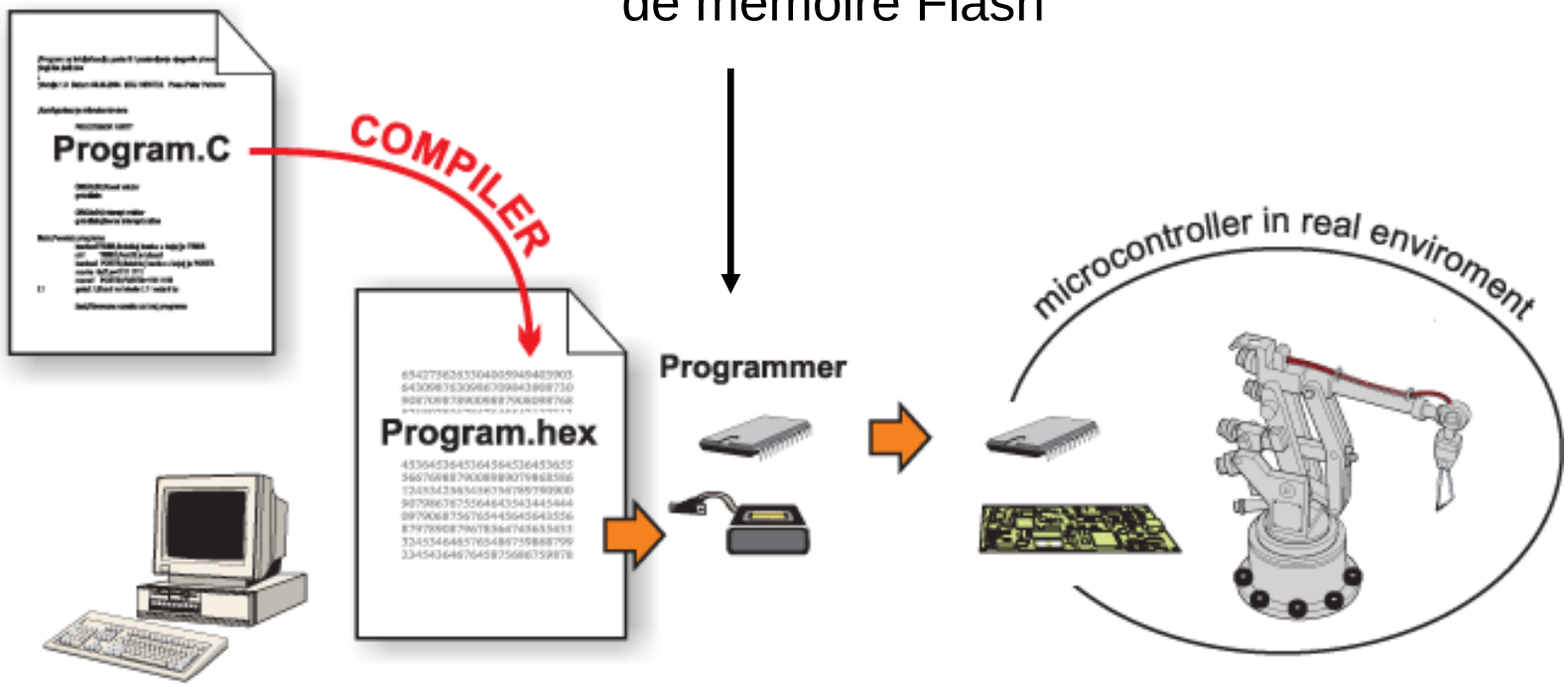
Compiled Program

```
:100000000428FF3FFF3FFF3F03138316860155304F
:10001000831286000A28FF3FFF3FFF3FFF3FFF3F5D
:04400E00F22FFFFF8F
:00000001FF
```

Executable Code of the program (HEX code)

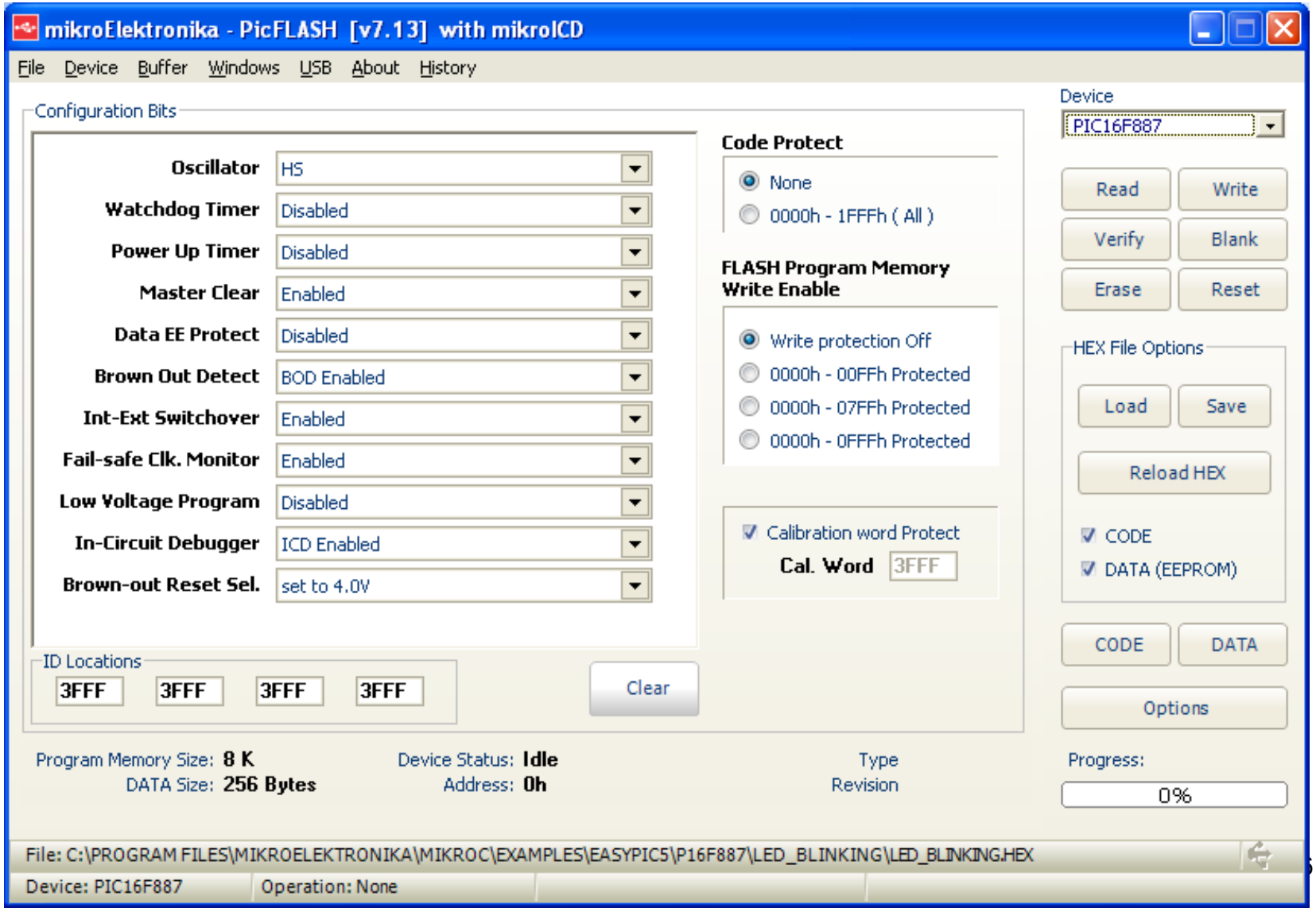
Le logiciel
Programmation de la mémoire programme (Flash)

Utilisation d'un logiciel de programmation de mémoire Flash



Le logiciel

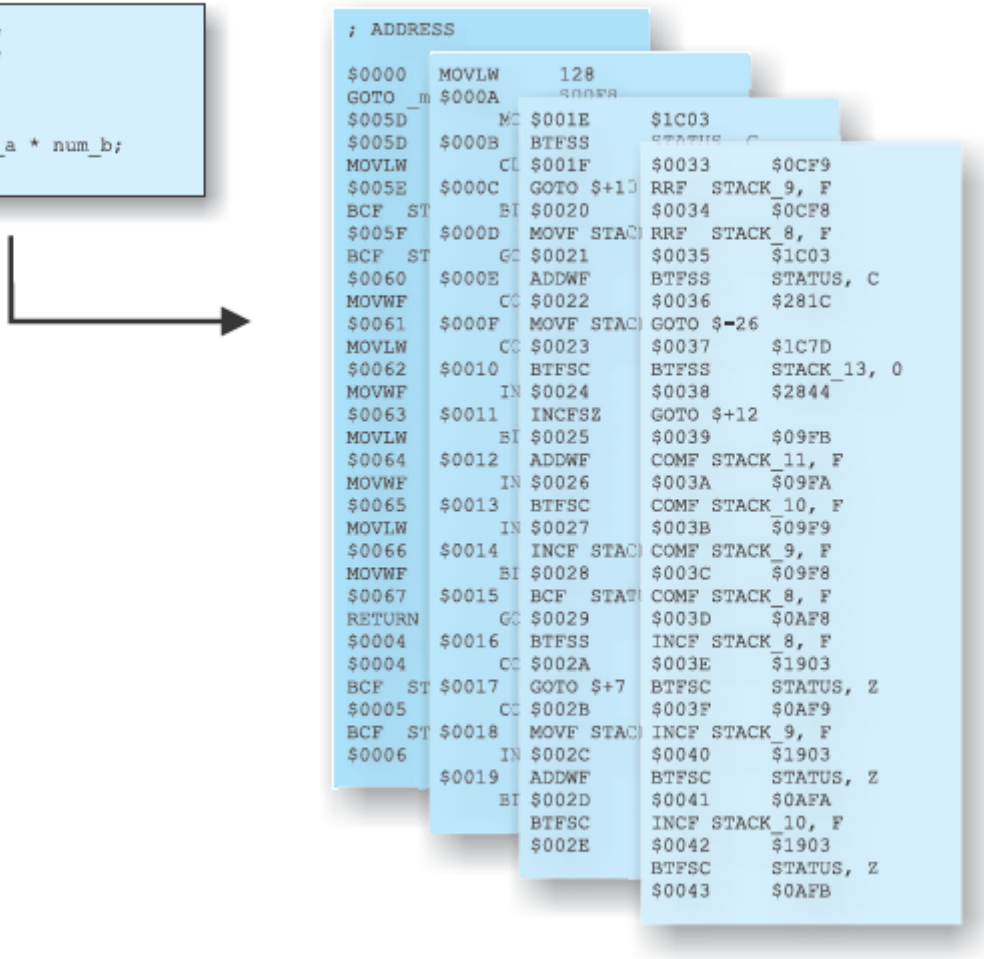
Programmation de la mémoire programme (Flash)



Le logiciel
Avantage du langage C

Program Written in C language

```
int num_a = 34;
int num_b = 14;
int result;

void main() {
  result = num_a * num_b;
}
```



Same program compiled into assembly code