



INTRODUCTION A L'INFORMATIQUE INDUSTRIELLE

CHAPITRE III

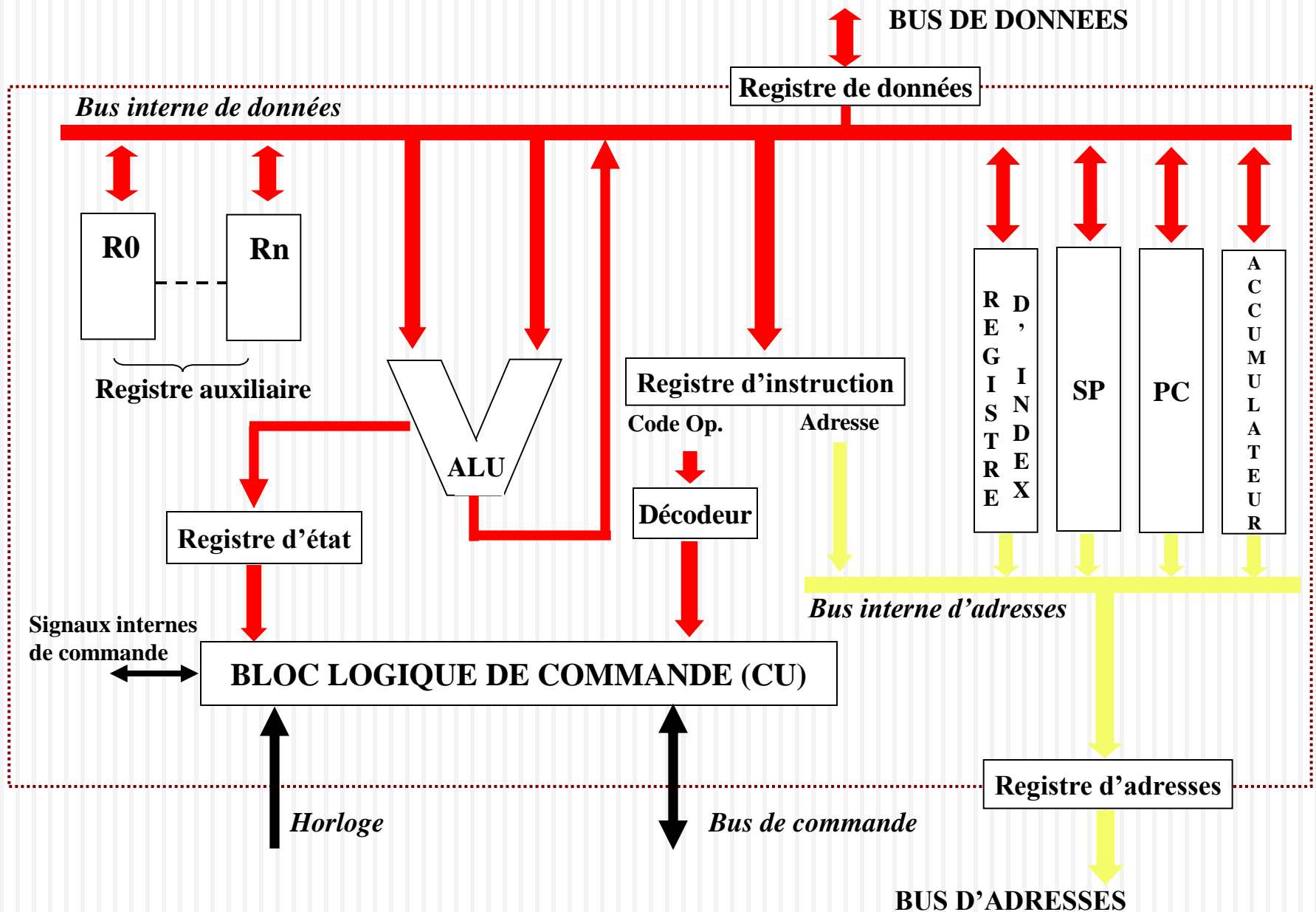
DU MICROPROCESSEUR AU MICROCONTROLEUR

Plan



- Le schéma interne d'un microprocesseur
- Le système minimal à base d'un microprocesseur
- Les Microcontrôleurs

Le schéma interne d'un microprocesseur



Compteur de programme

- **Il est constitué par un registre dont le contenu est initialisé avec l'adresse de la première instruction du programme.**
- **Dès le lancement du programme ce compteur contient l'adresse de la première instruction à exécuter :**
 - **soit par incrémentation automatique dans le cas où les adresses des instructions se suivent.**
 - **soit par chargement de l'adresse de branchement dans le cas de sauts programmés.**

ALU :

C'est un circuit complexe qui assure les fonctions:

- arithmétiques: addition et soustraction**
- logiques: ET, OU, OU exclusif**
- comparaison, décalage à droite ou à gauche, incrémentation, décrémentation, mise à 1 ou à 0 d'un bit, test de bit.**

Une ALU est constituée par un certain nombre de circuits tels que: complémenteur, additionneur, décaleur, portes logiques, ...

Bloc logique de commande:

- Il organise l'exécution des instructions au rythme d'une horloge.
- Il élabore tous les signaux de synchronisation internes ou externes (bus des commandes) du microprocesseur

Registre et décodeur d'instructions:

- Chacune des instructions à exécuter est rangée dans le registre instruction dont le format est 24 bits.
- Le premier octet (8 bits) est toujours le code de l'opération que le décodeur d'instruction doit identifier.

Pointeur de pile ou stack pointeur:

- **C'est un registre compteur de 16 bits qui contient l'adresse du sommet de la pile.**
- **La pile est externe au microprocesseur.**
- **C'est une partie de la mémoire RAM.**
- **Elle est utilisée pour sauvegarder les contenus des différents registres, lors de l'appel à un sous-programme ou lors de la gestion d'une interruption, par exemple.**

Registre de données

- **Ce registre de 8 bits est un registre tampon qui assure l'interfaçage entre le microprocesseur et son environnement ou inversement.**
- **Il conditionne le bus externe ou le bus interne des données.**

Registre d'adresses

- **Ce registre de 16 bits est un registre tampon qui assure l'interfaçage entre le microprocesseur et son environnement.**
- **Il conditionne le bus externe des adresses.**

Accumulateur :

- **Un accumulateur est un registre de travail de 8 ou 16 bits qui sert:**
 - **à stocker un opérande au début d'une opération arithmétique et le résultat à la fin de l'opération.**
 - **à stocker temporairement des données en provenance de l'extérieur du microprocesseur avant leur reprise pour être rangées en mémoire.**
 - **à stocker des données provenant de la mémoire ou de l'UAL pour les présenter vers l'extérieur du microprocesseur.**

Registre d'état :

- **Chacun de ces bits est un indicateur dont l'état dépend du résultat de la dernière opération effectuée.**
- **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 comme indicateur :

- ▶ **retenue**
- ▶ **retenue intermédiaire**
- ▶ **signe**
- ▶ **débordement**
- ▶ **zéro**
- ▶ **parité**

(carry : C)

(Auxiliary-Carry : AC)

(Sign : S)

(overflow : OV ou V)

(Z)

(Parity : P)

Registre d'index :

- **Le contenu de ce registre de 16 bits est une adresse.**
- **Il est utilisé dans le mode d'adressage indexé**

Registres auxiliaires :

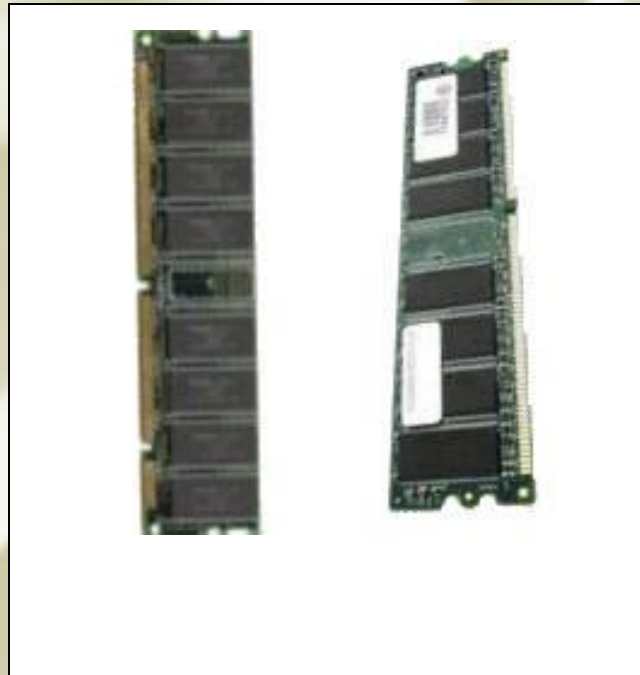
Ils permettent de stocker le résultat des instructions exécuter par l'ALU

Le système minimum à base d'un microprocesseur

Ces trois éléments vont communiquer entre eux par l'intermédiaire de 3 BUS



Unite centrale
(CPU)



Interface
Entrées/Sorties

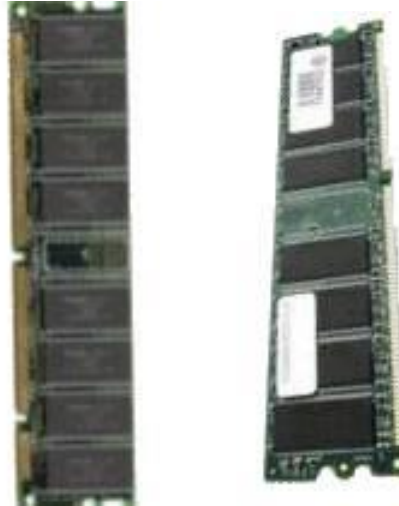
Ensemble de fils sur lesquels le CPU fournit l'adresse de la case mémoire sélectionnée.

Bus d'adresses

Il est unidirectionnel



Unité centrale
(CPU)



Interface
Entrées/Sorties

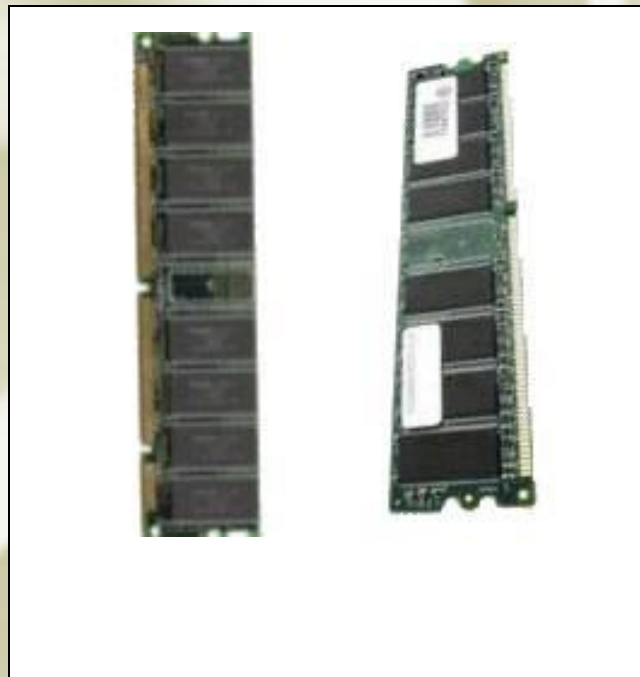
Sur ce bus se déplacent les données à traiter.

Il est Bidirectionnel.

Son nombre de lignes est égal à la capacité de traitement du microprocesseur.



Unite centrale
(CPU)



Interface
Entrées/Sorties

Bus de données

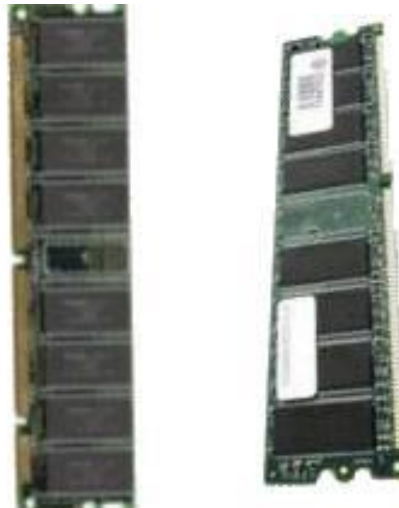
Ce bus véhicule des signaux relatifs aux :

Interruptions, commande de lecture/écriture,
des horloges de transfert, validation mémoire.

Le CPU indique ce qu 'il est en train de faire.



Unite centrale
(CPU)



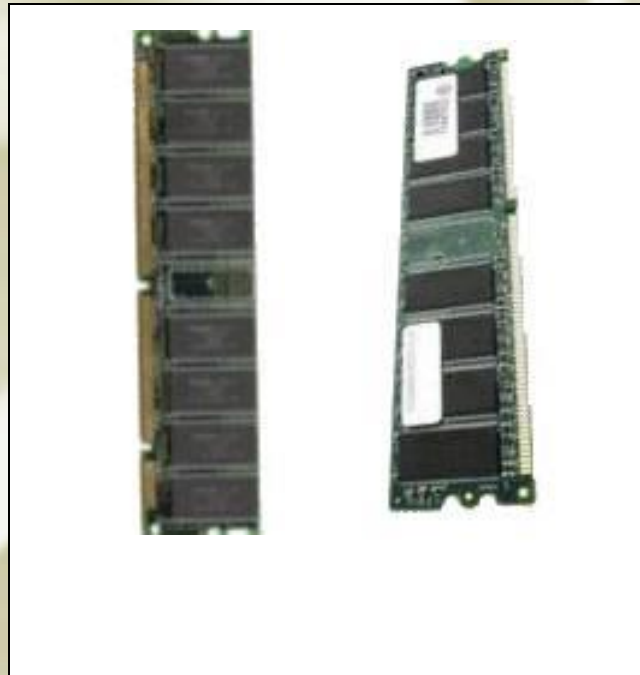
Interface
Entrées/Sorties

Bus de commande

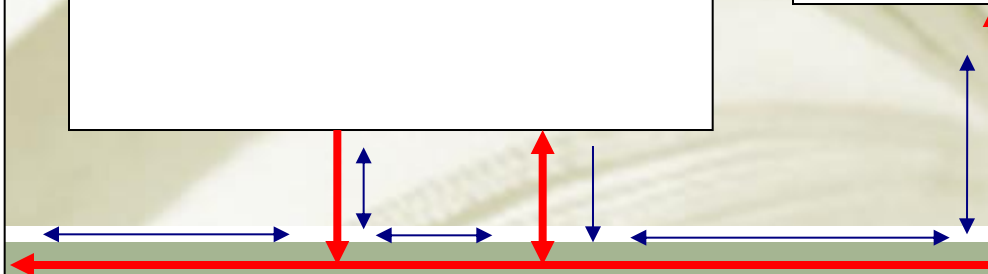
Dans quels boîtiers vont aller les données ?



Unité centrale
(CPU)



Interface
Entrées/Sorties

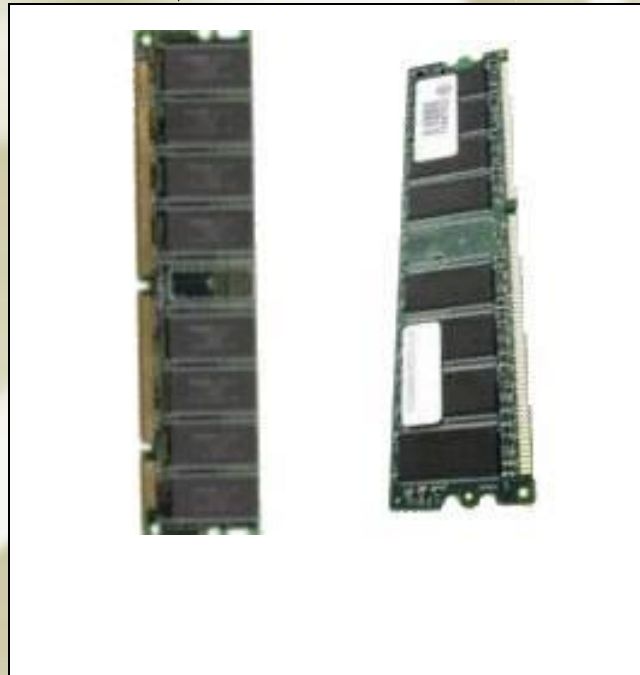


Décodeur d'adresses

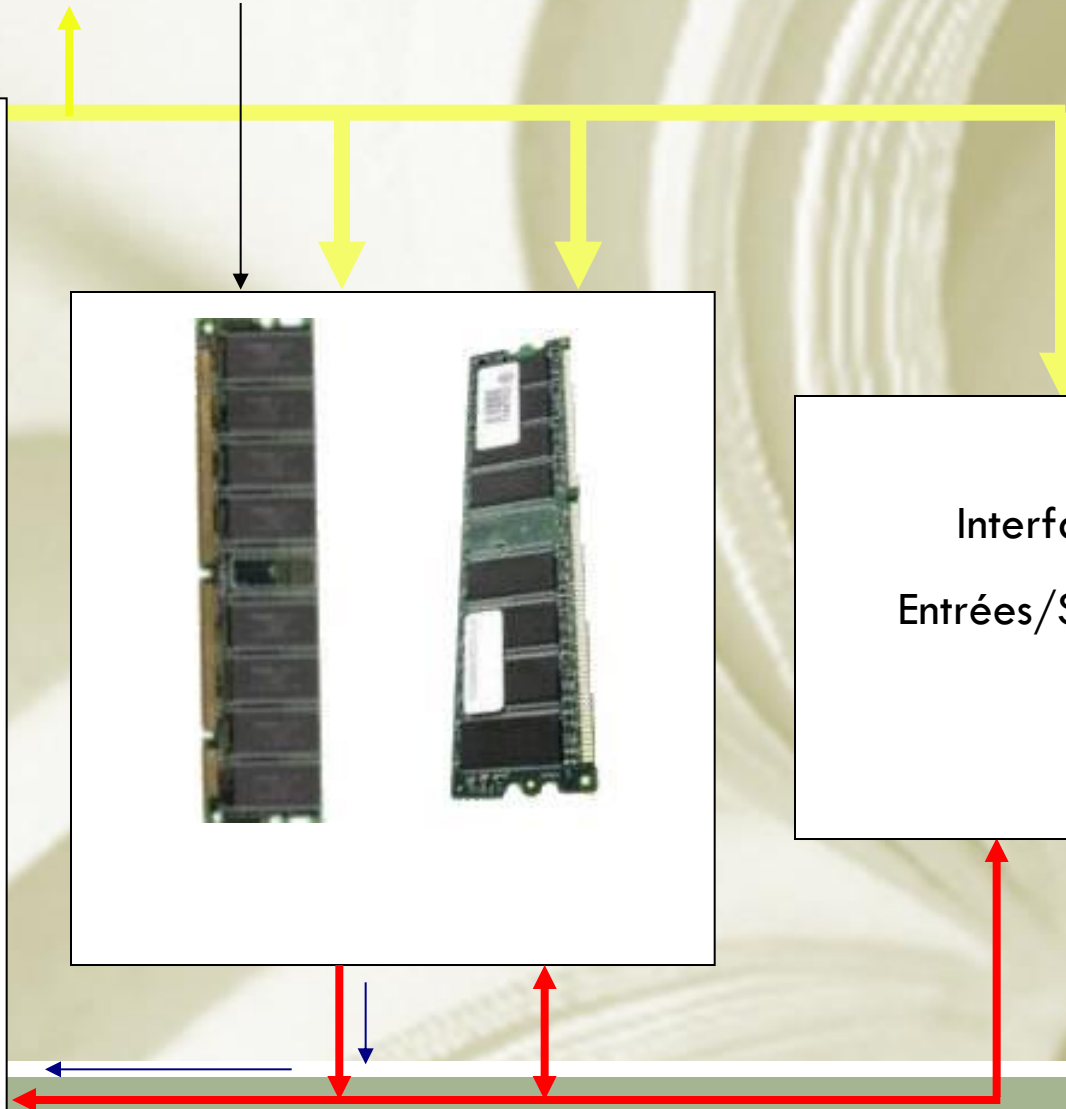
Le décodeur d'adresses
sélectionne la ROM



Unité centrale
(CPU)



Interface
Entrées/Sorties

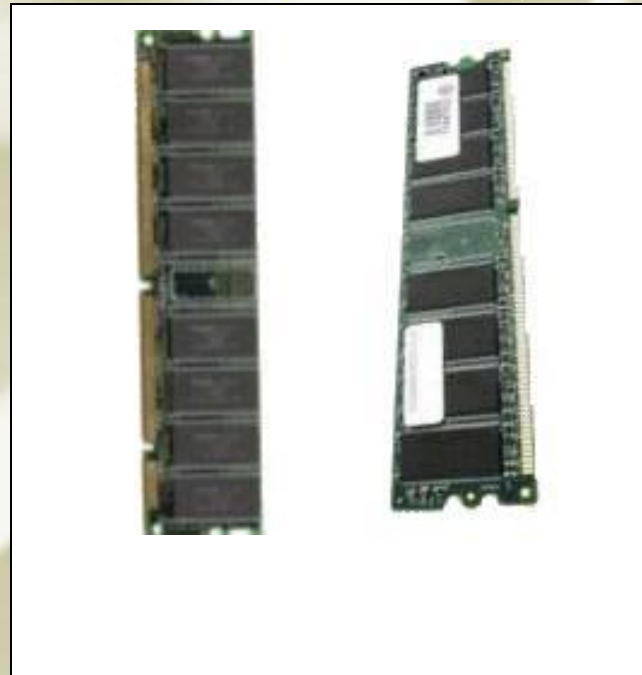


Décodeur d'adresses

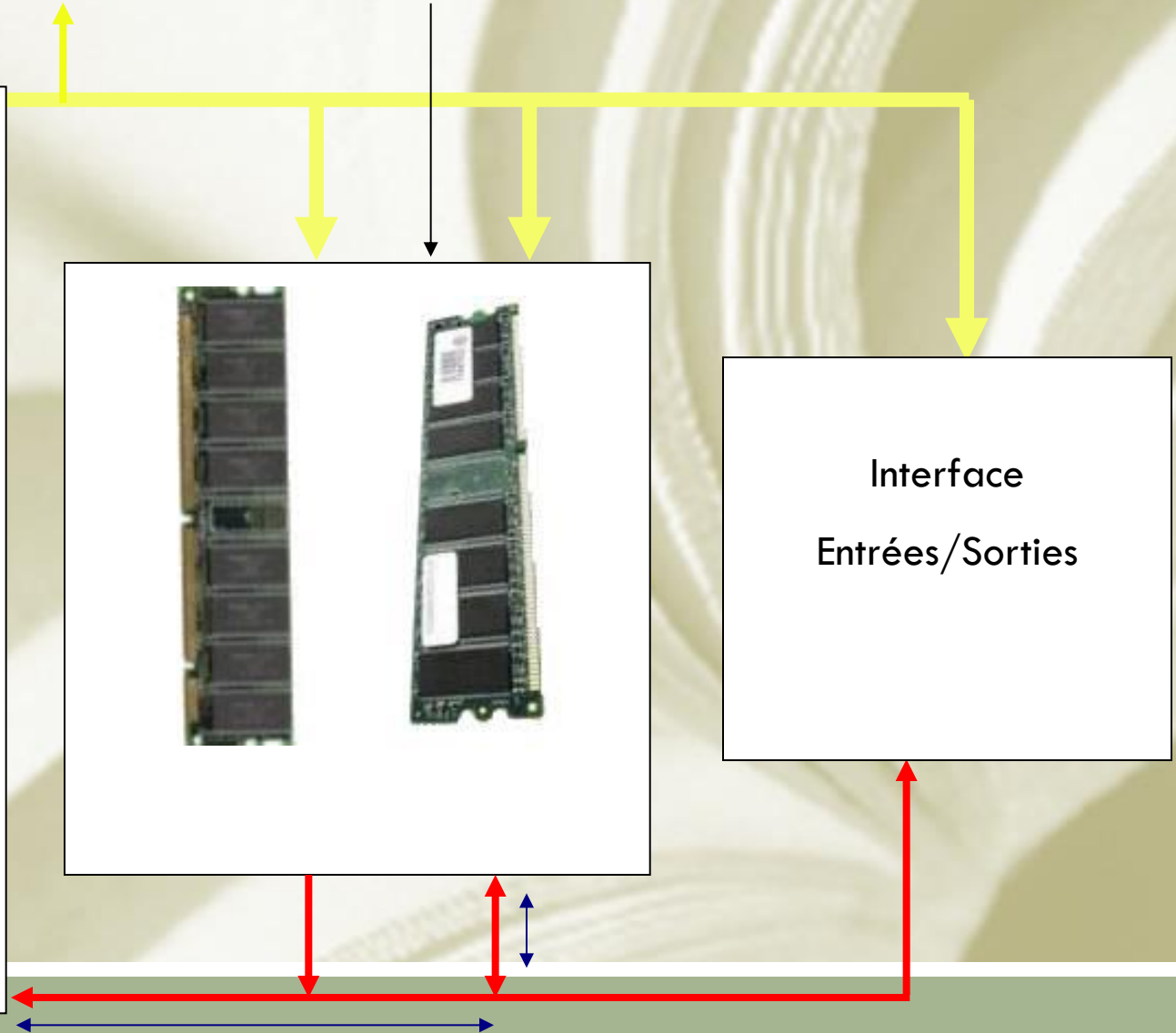
Le décodeur d'adresses
sélectionne la RAM



Unité centrale
(CPU)



Interface
Entrées/Sorties

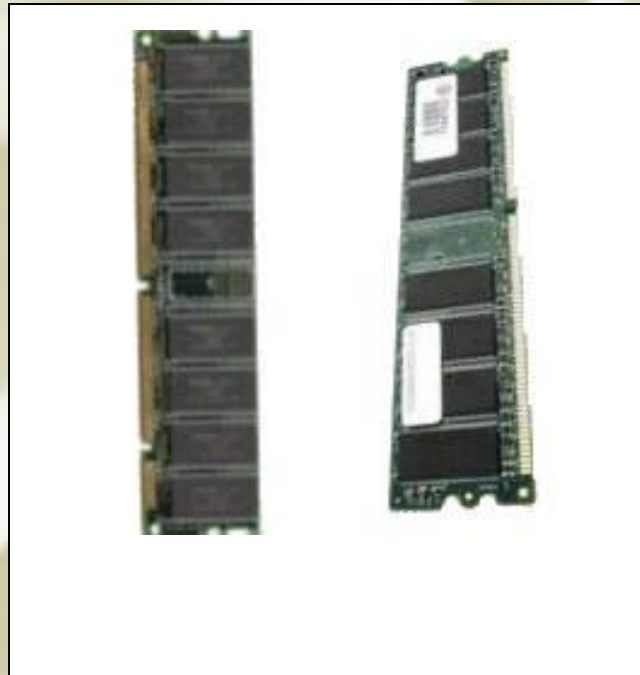


Décodeur d'adresses

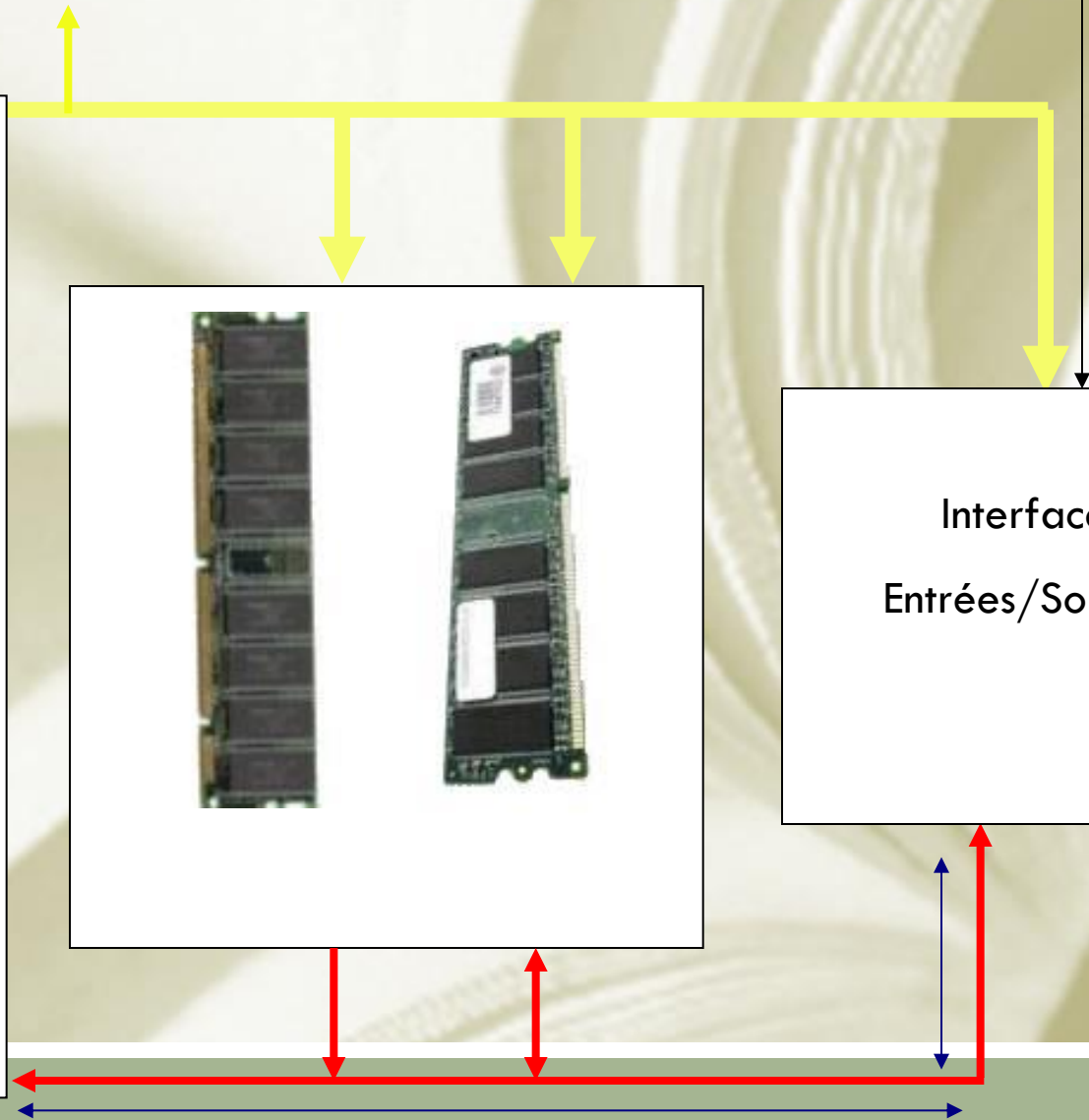
Le décodeur d'adresses
sélectionne l'I/O



Unité centrale
(CPU)



Interface
Entrées/Sorties



EXEMPLE

A la page suivante, on trouve le schéma structurel du système minimum
Dont les différents boîtiers se distinguent par leurs couleurs:

La CPU (rouge)

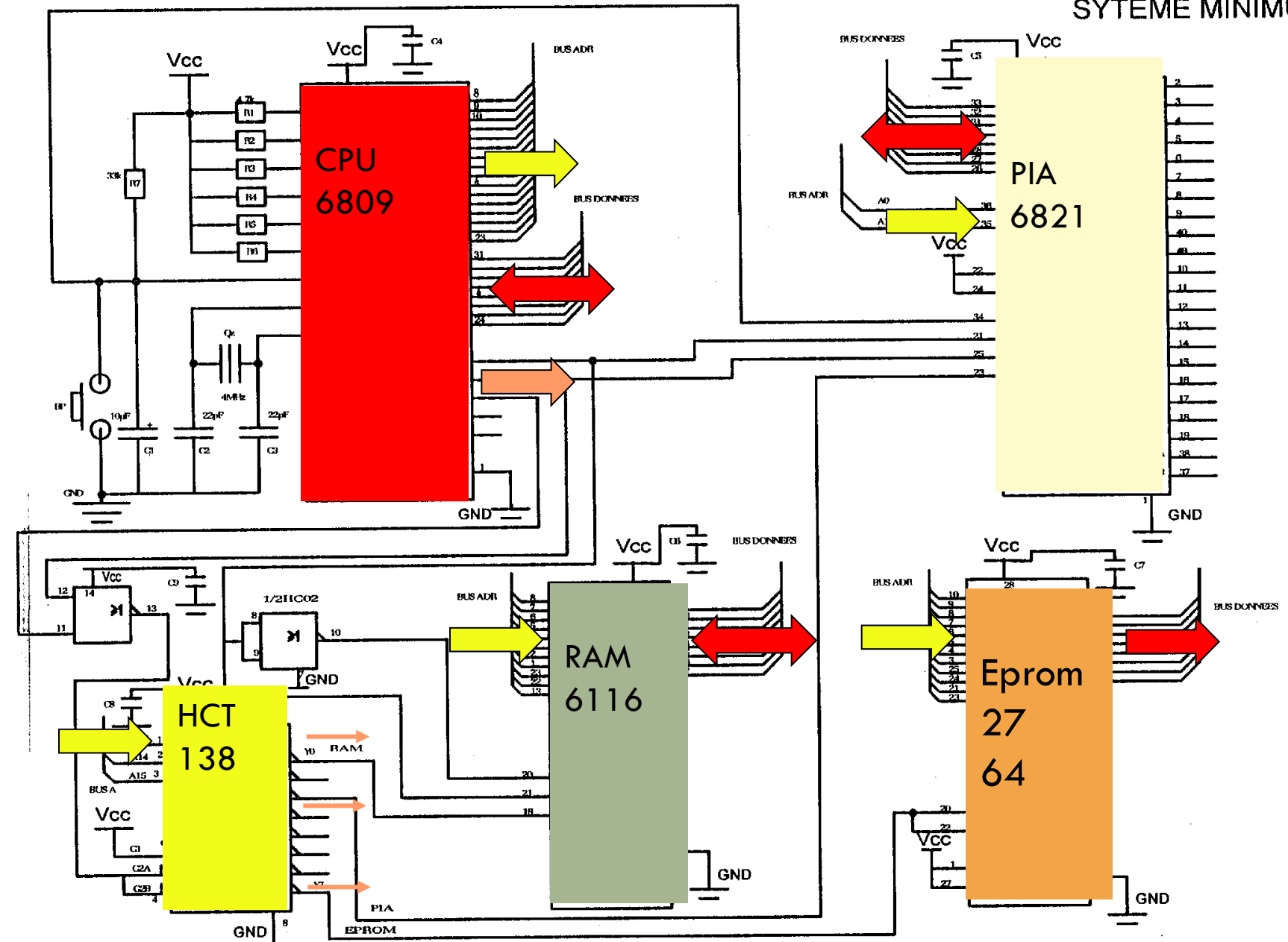
La ROM (bleu)

La RAM (vert)

Le circuit I/O (Gris)

Le décodeur d'adresses. (Jaune)

SYTEME MINIMUM



Système minimum

Les microcontrôleurs

PIC 16F..... Microchip

Sommaire

- *Mise en situation: - Evolution technologique
- Analyse fonctionnelle*
- *Microcontrôleur ou Microprocesseur?*
- *Architecture interne: Von Neumann ou Harvard?*
- *Les registres internes*
- *La base de temps*
- *Le jeu d'instructions et les modes d'adressage*

Mise en situation

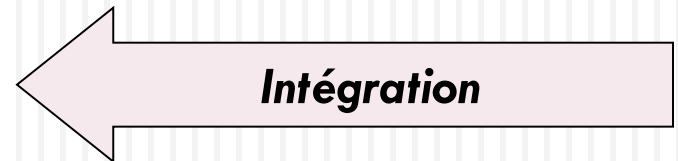
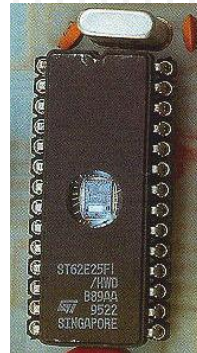
L'évolution des produits domestiques (ou industriels) rend compte d'un phénomène directement lié à l'***évolution des technologies***:

Evolution technologique

- Progrès de la **miniaturisation**.
Les téléphones portables en sont un exemple très actuel.

Mais cette miniaturisation ne peut se faire sans une **évolution de la technologie** utilisée.

- Progrès de l'**'intégration**.
Le nombre de structures intégrées à un seul composant est de plus en plus important.
Le nombre de circuits utilisés est ainsi réduit.



Microtechnologie

On comprendra aisément qu'un **système microprogrammé** tel que le téléphone portable ne peut être géré par un système minimum à microprocesseur 6809: ***trop encombrant!***

La solution est alors de remplacer le système minimum par ***un seul circuit:***
Le ***microcontrôleur.***

- Microprocesseur
- RAM
- EPROM
- PIA
- Décodeur

Solo:

- Microcontrôleur

Amplification

On retrouve ainsi les microcontrôleurs **PIC** dans de nombreuses applications **industrielles** ou **domestiques**.

Prenons l'exemple d'une **télécommande infra-rouge**:

Saisie touche
FP1

Reconnaissance touche
Génération commande
FP2

Emission IR de
la commande
FP3

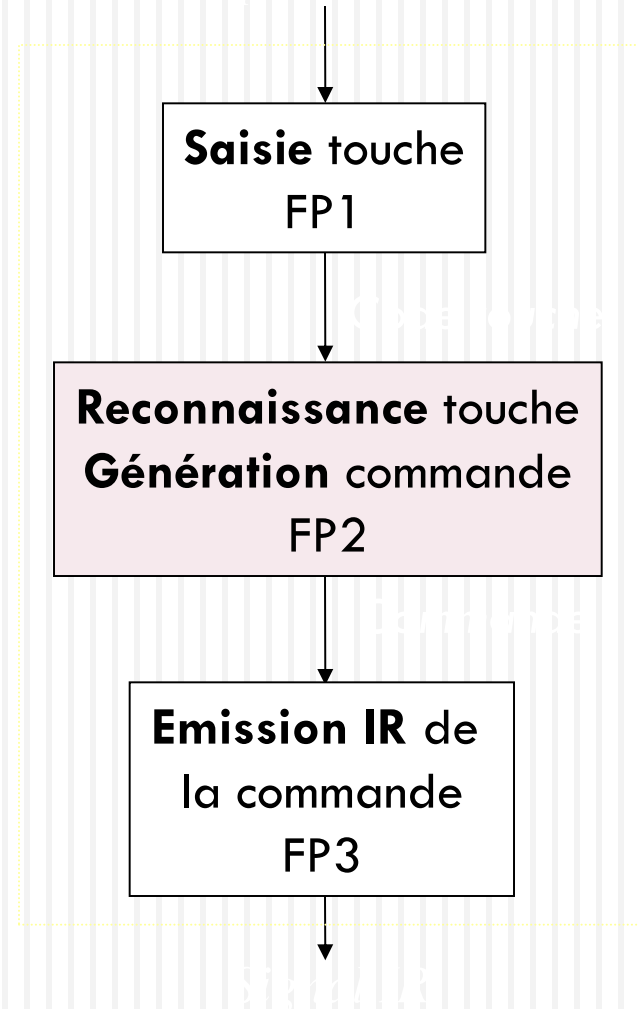
4.1.1.2 Fonctionnalité

La **fonction FP1** a pour rôle de prendre en compte l'appui sur une touche et de transmettre le code correspondant à la fonction FP2.

La **fonction FP2** a pour rôle d'identifier la touche à l'aide du « code touche » et de générer le signal de commande associé.

La **fonction FP3** se charge de convertir et émettre le signal de commande sous forme de signal infra-rouge.

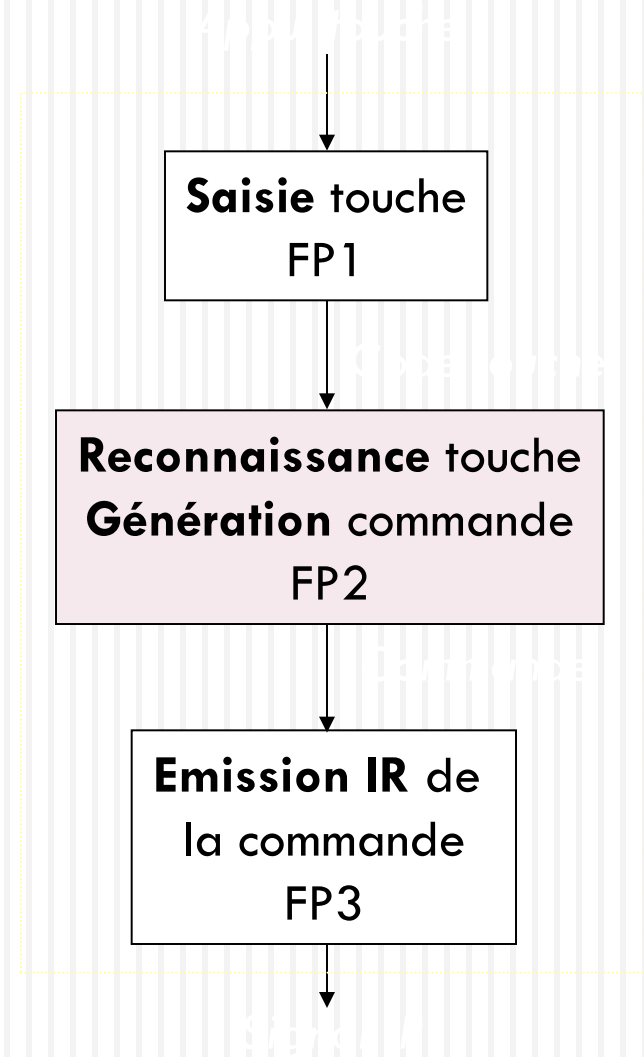
Architecture



4.1.2 Fonctionnalité

La **fonction FP2** « Reconnaissance touche et génération commande » est réalisée par une structure microprogrammée.

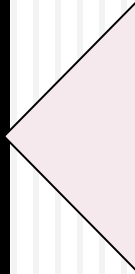
C 'est ici un **microcontrôleur PIC** qui se charge, par l 'exécution de son programme, de faire l 'acquisition du signal « code touche », et de générer de signal de commande correspondant.



Microcontrôleur ou Microprocesseur?

Suivant le type d 'application envisagé, il est possible de faire appel à différents types de **structures microprogrammées**. Les plus répandues sont les suivantes:

- Le **microprocesseur**.
- Le **microcontrôleur**.



Ex: PC, système minimum à **6809** ,
Pentium4, DualCore...



Ex: **PIC**, 68HC11, Atmel AVR 16-bits..

Microcontrôleur

Un système à microprocesseur nécessite une grande **place matérielle** (nombreux circuits) ainsi qu'une bonne qualité de **connectique**.

Les microcontrôleurs permettent quant à eux de s'affranchir de ces contraintes puisque'ils **intègrent** en un seul circuit au moins toutes les ressources propres à un système minimum.

Ainsi, les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes:

- **Mémoire de programme.**

Les PIC 16Cxx se déclinent selon 3 versions de mémoire de programme:

- **ROM** (ou OTPROM),
programmable une seule fois.

Capacité: 512 à 2Kmots.

- **UVPROM**, effaçable par rayonnement UV.

Capacité: 512 à 4Kmots.

- **EEPROM**, effaçable électriquement.

Capacité: 1Kmots.

Ainsi, les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes:

- **Mémoire de programme.**
- **Mémoire de données.**

Les PIC 16Cxx disposent d'une **mémoire de données** (RAM) de capacité 25 à 192 octets.

Ainsi, les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes:

- **Mémoire de programme.**
- **Mémoire de données.**
- **Entrées/sorties.**

Les PIC 16Cxx proposent un certain nombre de **broches d'entrées/sorties** (12 à 33) permettant l'acquisition ou la transmission de signaux numériques.

Ainsi, les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes:

- **Mémoire de programme.**
- **Mémoire de données.**
- **Entrées/sorties.**

Et éventuellement:

- **Port série.**

Certains PIC 16Cxx possèdent 1 ou 2 **ports série** permettant la transmission série d'informations numériques.

Ainsi, les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes:

- **Mémoire de programme.**
- **Mémoire de données.**
- **Entrées/sorties.**

Et éventuellement:

- **Port série.**
- **Convertisseur CAN.**

Certains PIC 16Cxx possèdent en ressource interne un **Convertisseur Analogique Numérique 8bits** permettant l'acquisition de 4 à 8 signaux analogiques différents.

Conclusion:

Les microcontrôleurs PIC 16Cxx sont des circuits complets et performants.

Ils s'appliquent complètement dans la mise en œuvre de systèmes microprogrammés simples.

Architecture interne

La majorité des structures microprogrammées utilisent une architecture classique appelée: ***Architecture Von Neumann.***

Architecture **Von Neumann:**
PC, **6809**, 68HC11...

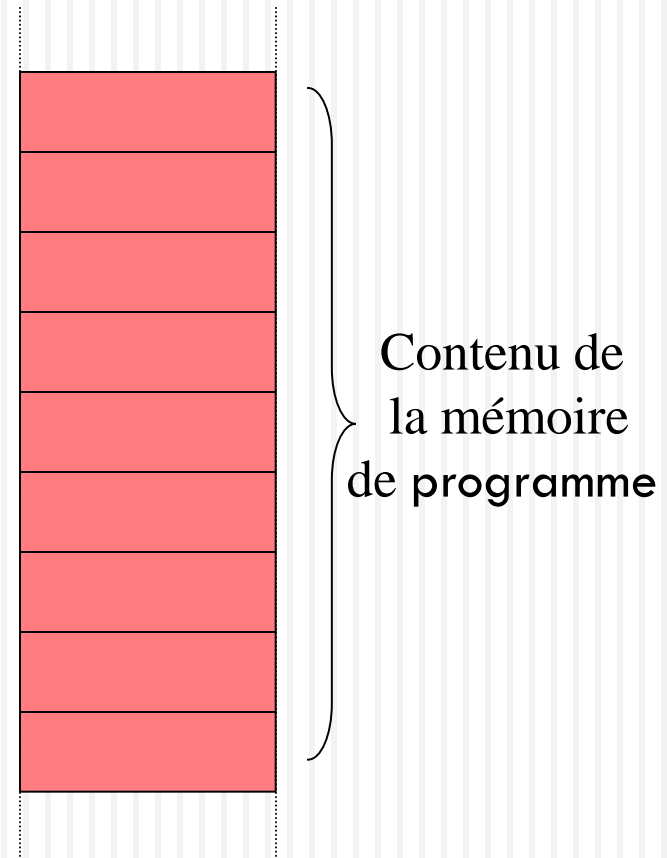
Les microcontrôleurs PIC ainsi que bien d'autres structures sont construites autour d'un autre type d'architecture: ***Architecture Harvard.***

Architecture **Harvard:**
PIC, DSP...

Architecture Von Neumann

Prenons le cas du système minimum à **6809**. Son architecture est de type ***Von Neumann***.

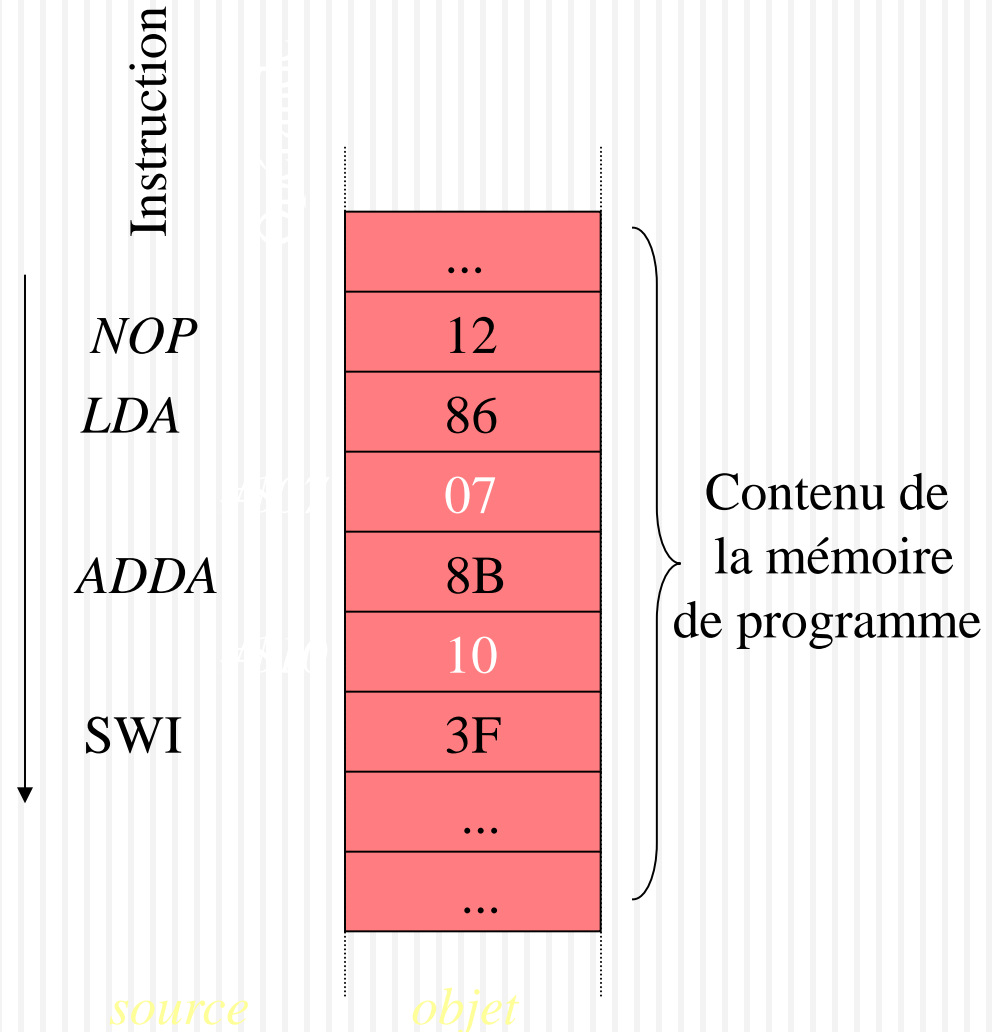
Sa ***mémoire de programme*** (EPROM) contient comme son nom l'indique le programme à exécuter.



Considérons l'exemple du programme source suivant.

Après assemblage, chaque *instruction* et chaque *opérande* codée sur *un octet* (8 bits) est rangée dans une case de la mémoire.

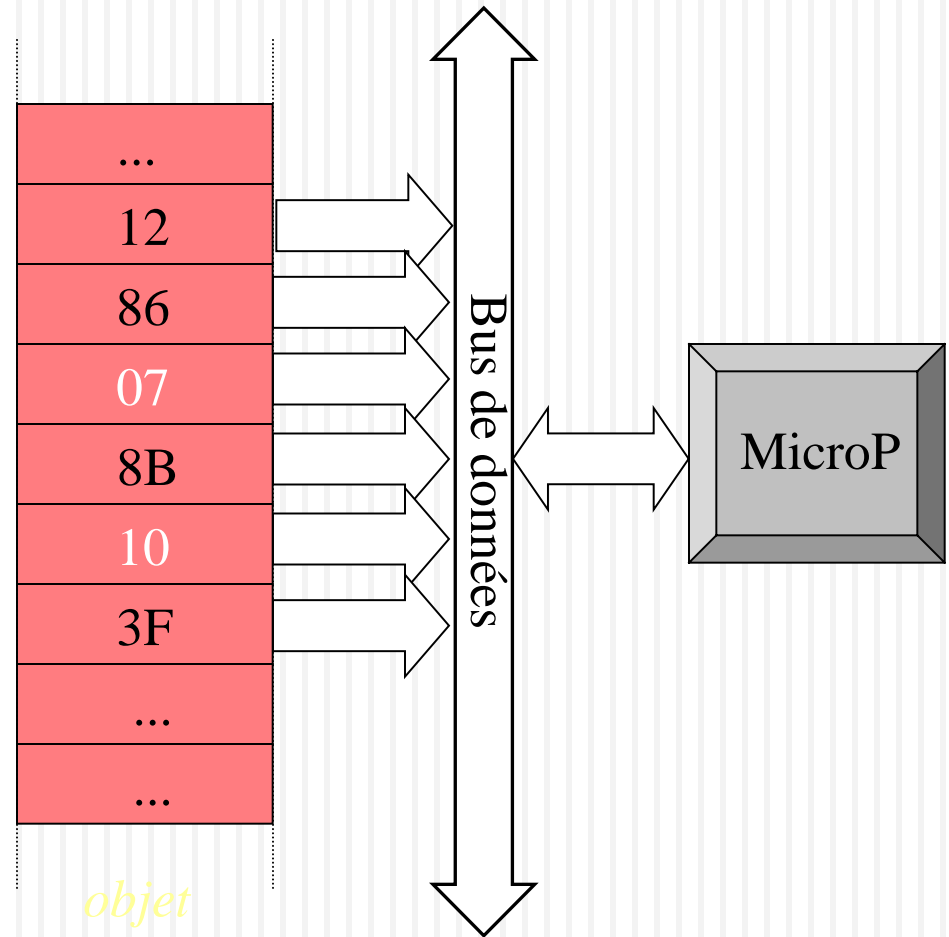
La mémoire contient donc *successivement* les instructions et les opérandes du programme .



Architecture Von Neumann

Afin d'exécuter le programme, le microprocesseur doit *lire dans l'ordre* le contenu de chacune des cases mémoires.

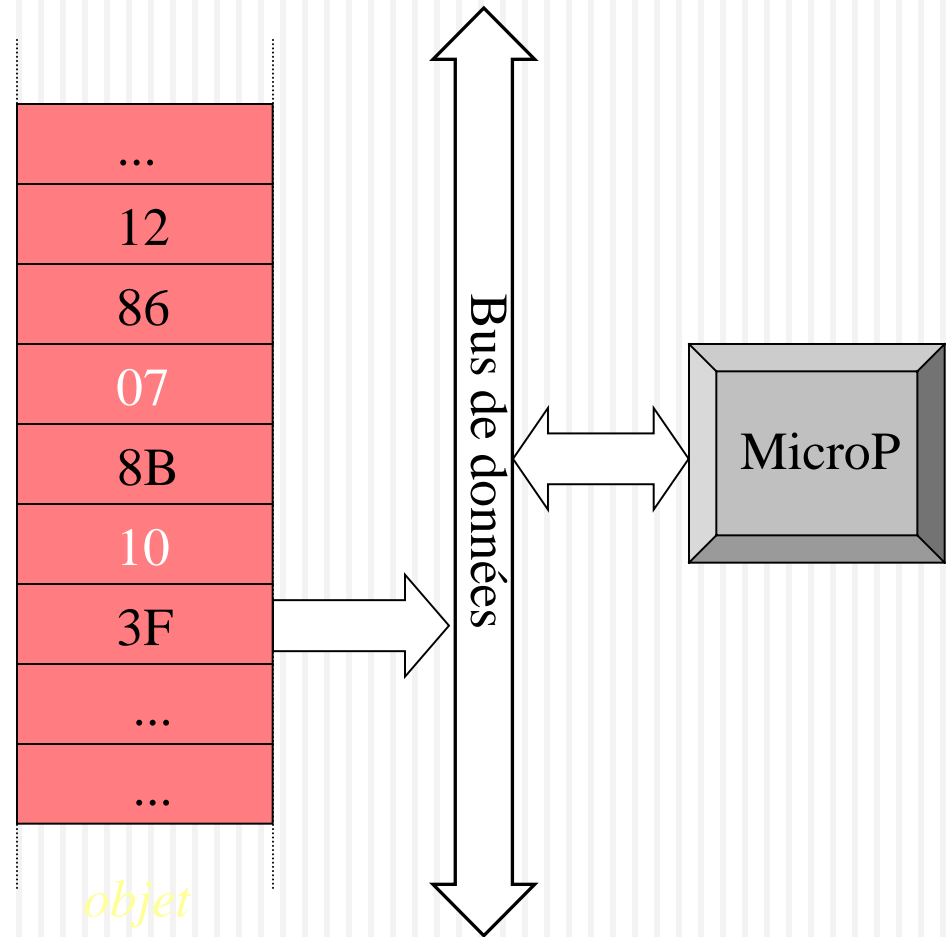
Pour cela, chacun des octets de la mémoire est acheminé vers le microprocesseur, via le *bus de données*.



Conclusion:

Dans le cas d'une architecture *Von Neumann*, le traitement d'une instruction et son opérande nécessite donc la lecture d'au moins *deux cases mémoires* (3 si l'opérande est codée sur deux octets).

Cela correspond à une durée de *2 ou 3 cycles machine*.



Architecture Harvard

Les microcontrôleurs **PIC** ont eux une architecture appelée **Harvard** qui présente de nombreux avantages.

Les différences avec les architectures Von Neumann résident essentiellement dans:

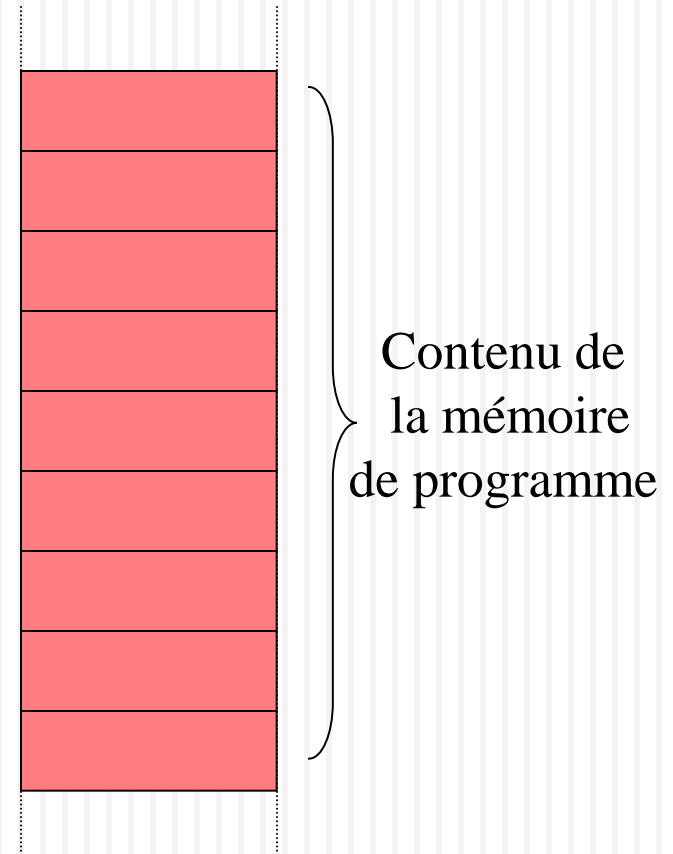
- la **mémoire de programme**
- les **bus**.

Architecture PIC

La *mémoire de programme* des **PIC** contient bien entendu le programme à exécuter.

Comme précédemment, ce programme est composé d'*instructions* et d'*opérandes*.

Cependant, une case mémoire peut ici contenir à la fois une instruction *et* son opérande.



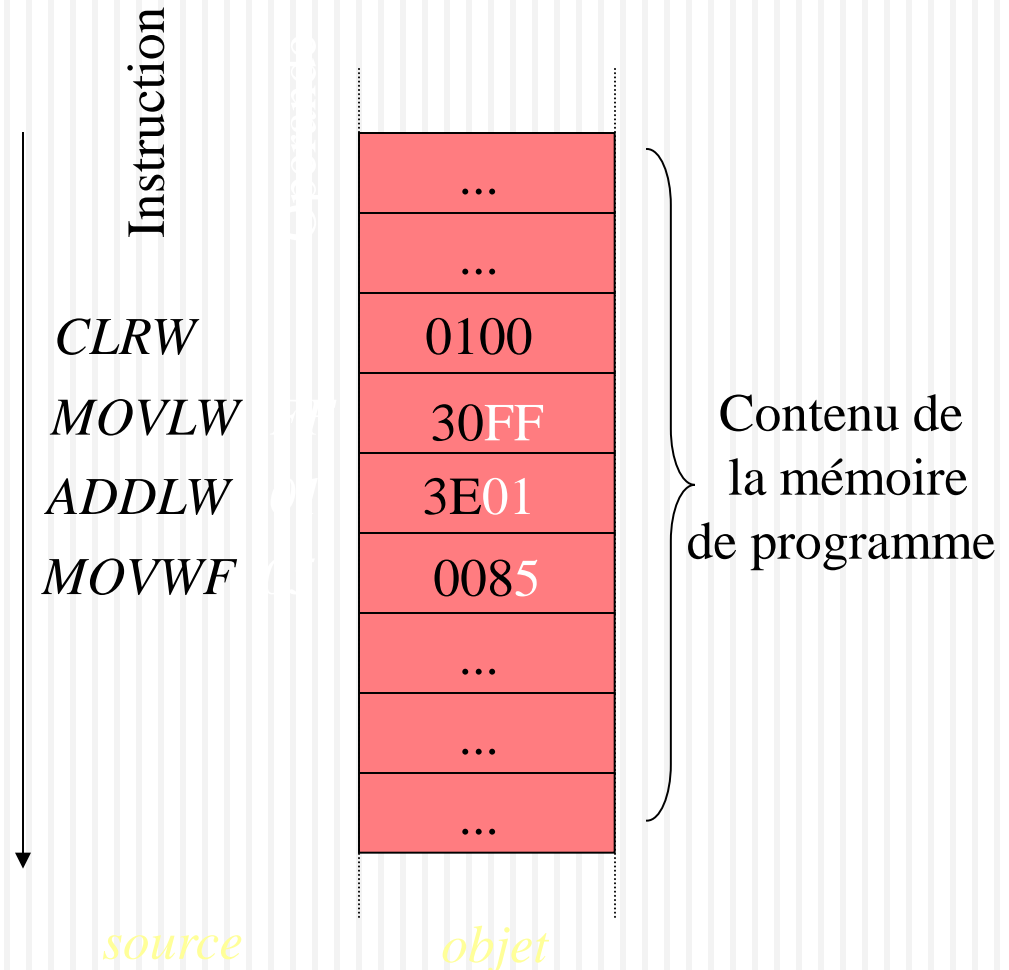
Chapitre 10 : Mémoire

Considérons l'exemple du programme source suivant.

Après assemblage, chaque *instruction* et son *opérande* sont codées sur *un mot binaire* (12 ou 14 bits) puis rangées dans une case mémoire.

Chaque cas de la mémoire contient donc:

- l'*instruction* à exécuter.
- L '*opérande* associée (non obligatoire).

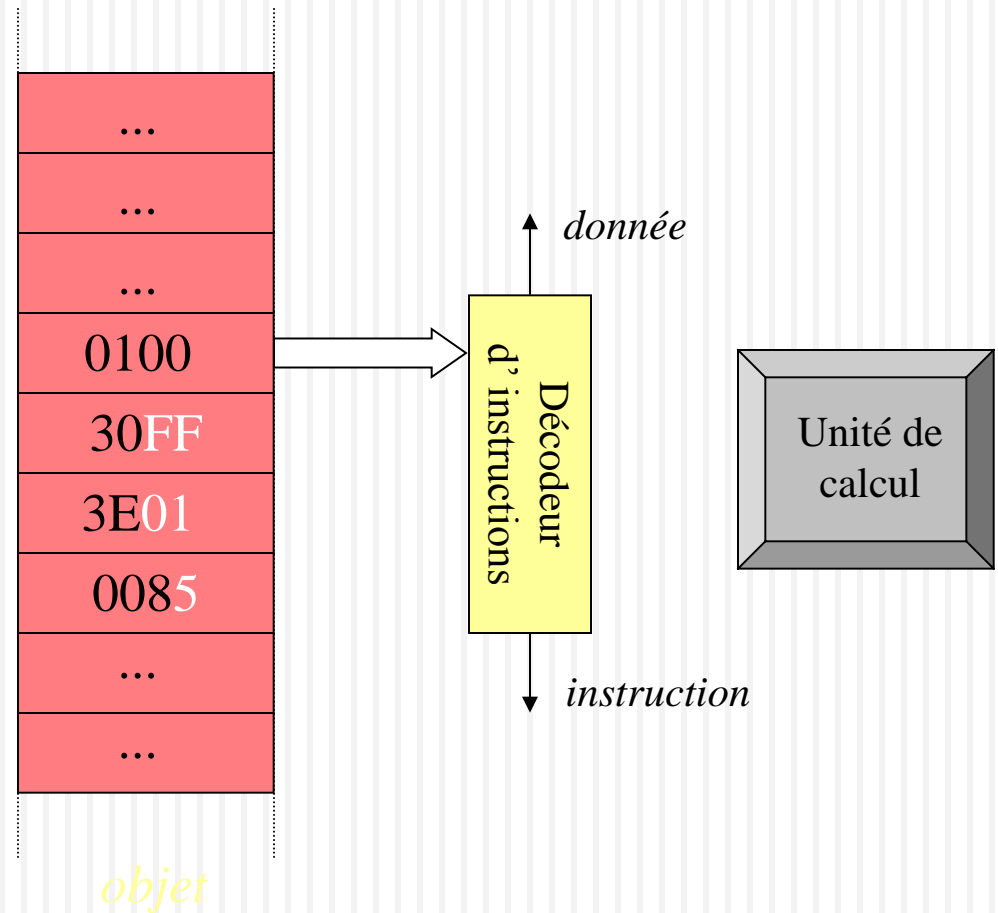


Architecture d'un ordinateur

Afin d'exécuter le programme, l'unité de calcul doit ensuite *lire* le contenu de chacune des cases de la mémoire.

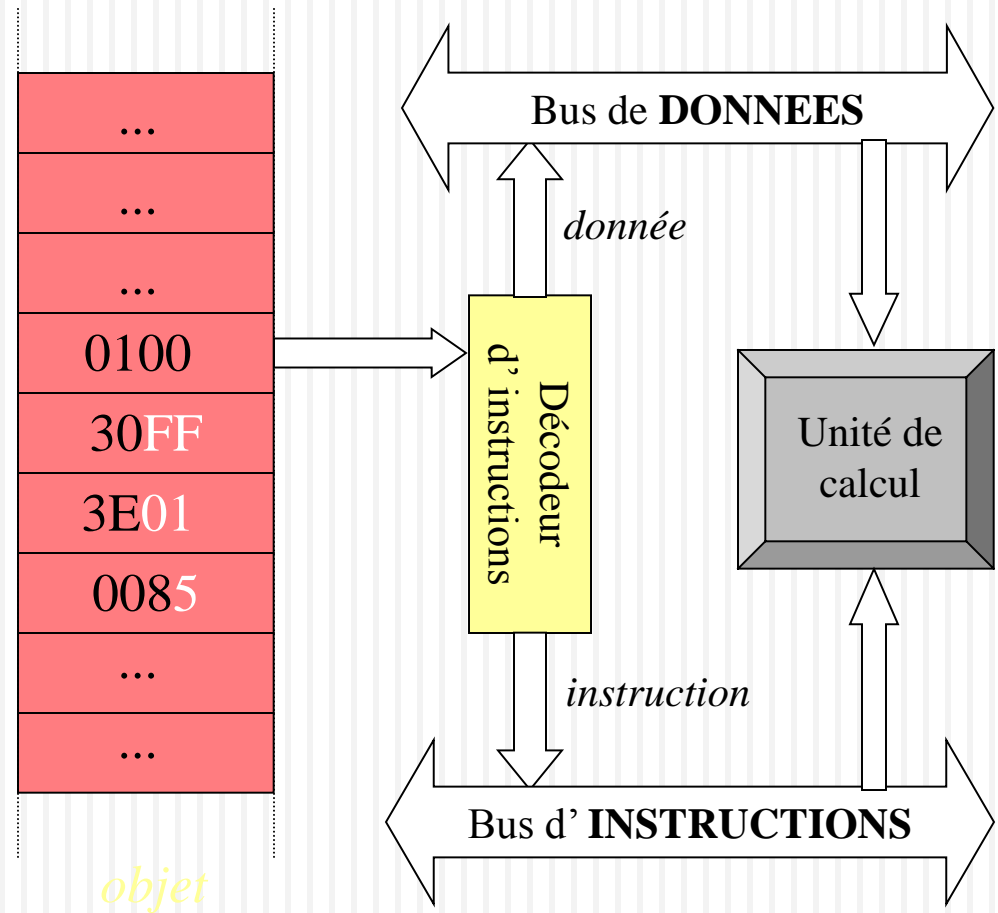
Chaque mot binaire contenu dans la mémoire de programme est alors acheminé vers un *décodeur d'instructions*.

Le rôle de ce décodeur est de *séparer* pour chacun des mots binaires, l'*instruction* et la *donnée* (opérande).



Architecture Harvard

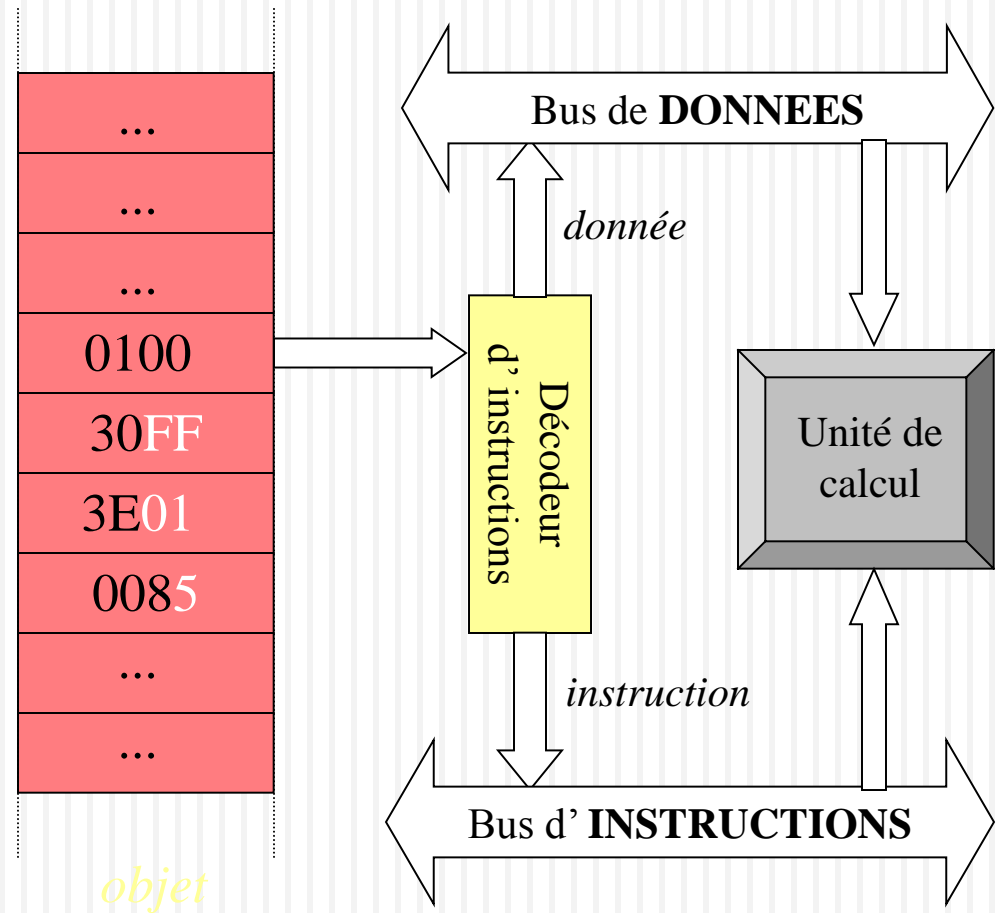
Les *instructions* et les *données* sont ensuite acheminées *simultanément* vers l'unité de calcul par l'intermédiaire de *deux bus différents*.



Conclusion:

Dans le cas de l'architecture **Harvard** que possèdent les PIC, la lecture d '**une seule case mémoire** permet le traitement entier d 'une instruction et de son opérande.






Un seul cycle machine est donc nécessaire.



Les registres internes

Selon la version de *PIC 16Cxx* utilisée, le nombre de *registres internes* au circuit est différent.

Ainsi, les registres présentés ci-après sont les plus couramment utilisés:

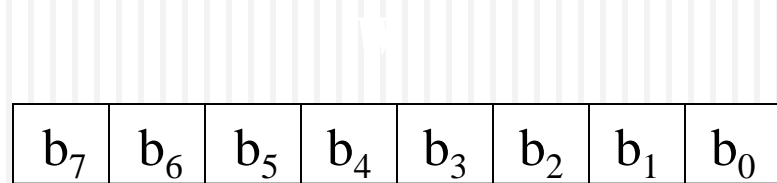
- Registre de *travail*: 
- Registres d '*E/S*: 
- Registres de *direction*: 
- Registre d '*état*: 
- Registre *Compteur Programme*: 

Registre de travail W

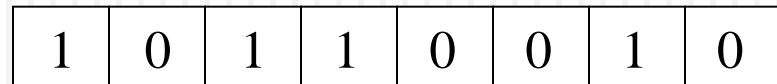
Le registre de travail **W** est un **registre 8 bits** destiné à la manipulation générale des données.

Il peut donc contenir une donnée de 8 bits que l'on appelle ici un **littéral**.

Le registre W peut être comparé aux registres A ou B du 6809.



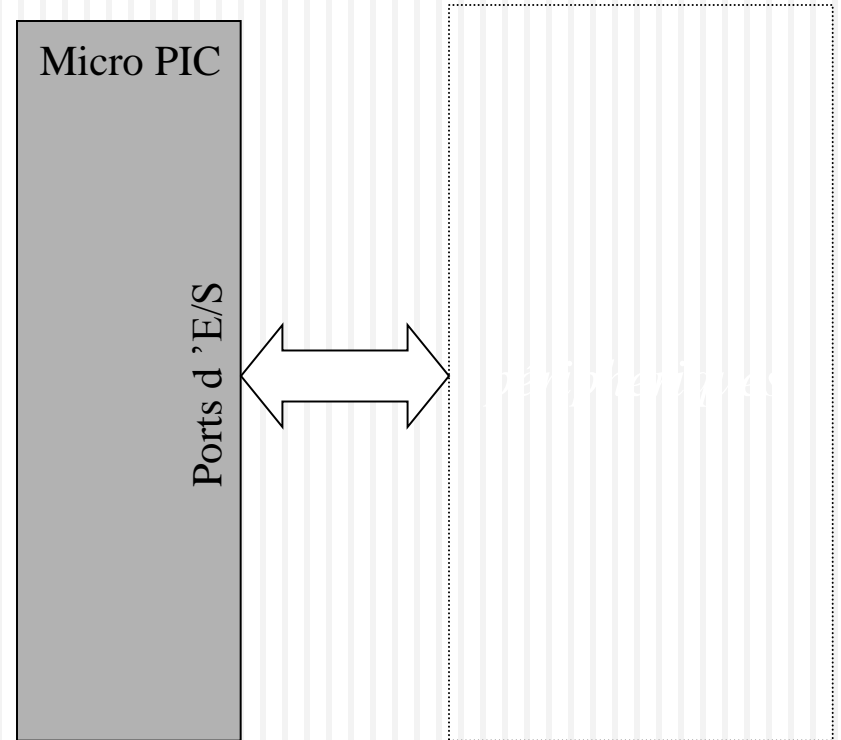
Ex:



Le **littéral** chargé dans le registre de travail a pour valeur hexadécimale B2.

Registres d'E/S PORT

Les microcontrôleurs **PIC** peuvent *recevoir ou transmettre* des informations avec des *périphériques extérieurs* par l'intermédiaire de leurs *ports d'E/S*.

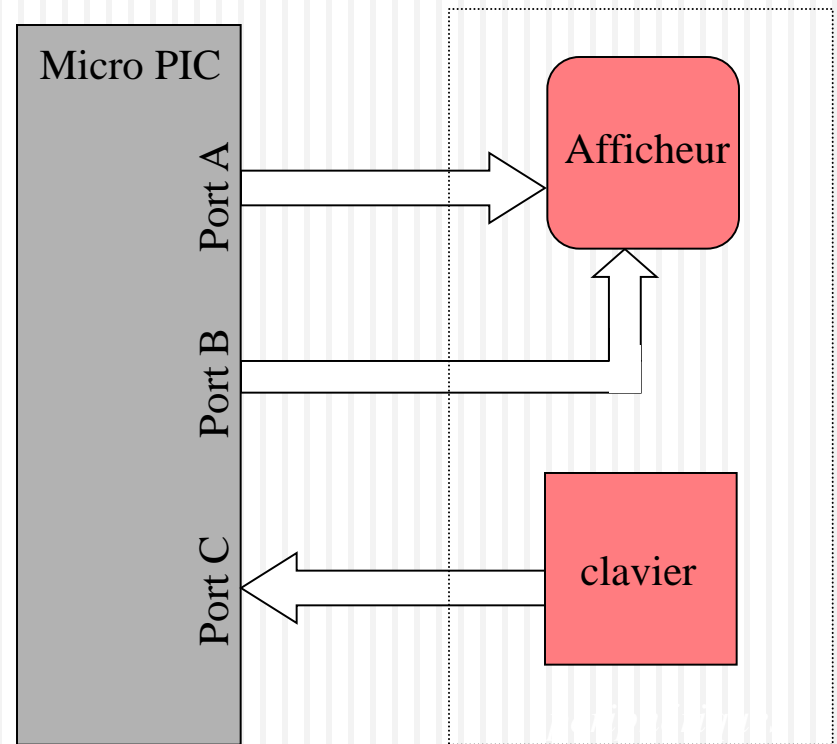


Exemple d'E/S PORT

Suivant la version utilisée, les circuits proposent **2 ou 3 ports d'E/S** différents.

Dans l'exemple suivant, le port C est utilisé pour **recevoir** des informations provenant d'un clavier.

Les ports A et B sont eux utilisés pour **transmettre** les données à afficher.



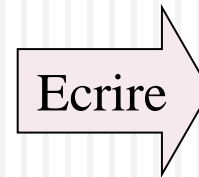
Registres I/O PORT

L'utilisation des registres est ainsi la suivante:

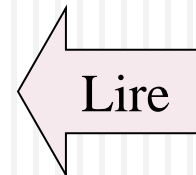
- Pour *transmettre une donnée* sur un port, il faut **ECRIRE** la donnée dans le *registre PORT* correspondant.

- Pour *recevoir une donnée* sur un port, il faut **LIRE** la donnée dans le *registre PORT* correspondant.

Donnée à transmettre



Donnée reçue



Registres I/O PORT

Remarque 1:

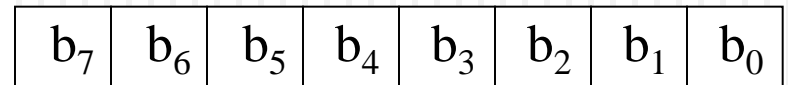
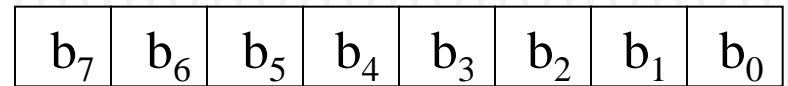
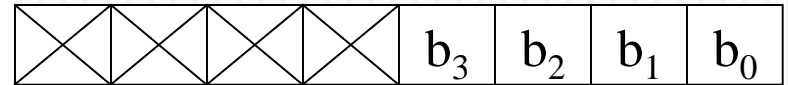
Les registres PORTA, PORTB et PORTC sont analogues aux registres ***ORA et ORB du 6821.***

Registres E/S PORT

Remarque 2:

Sur les PIC 16Cxx, le **port A** du circuit ne présente en fait que **4 broches d 'E/S**. Le registre correspondant (8 bits) n 'a donc d 'actifs que les bits **b_0 à b_3** .

Les ports **B et C** possèdent eux bien **8 lignes d 'E/S**. Les registres correspondant ont donc les **8 bits actifs**.



Registres de direction *TRIS*

Les registres de direction ***TRIS*** (***8 bits***) sont directement liés aux registres PORT.

Le rôle des registres ***TRIS*** est de programmer chacune des ***lignes*** des ports soit en ***entrée***, soit en ***sortie***.

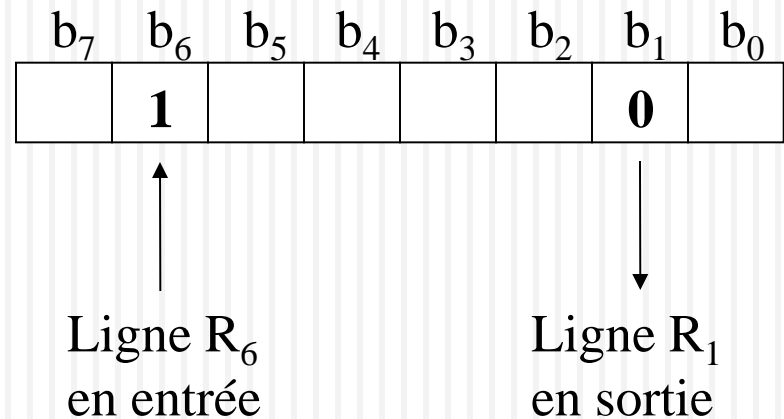
Les différentes broches (lignes) d'un même port peuvent donc avoir un rôle différent:

transmettre ou ***recevoir*** une valeur logique (« 0 » ou « 1 »).

Exemple de direction TRIS

La programmation des registres **TRIS** est la suivante:

- La mise à « **0** » du bit programme la ligne correspondant en *sortie*.
- La mise à « **1** » du bit programme la ligne correspondante en *entrée*.



Programme de direction TRIS

Exemple:

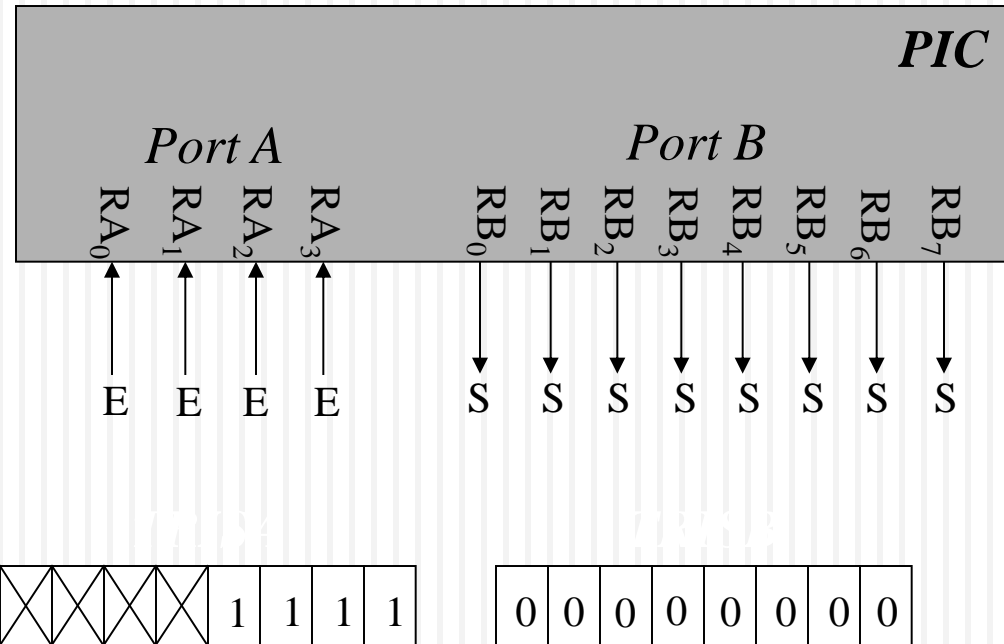
Programmons le *port A en entrée* et le *port B en sortie*.

Remarque: le port A ne possède que 4 lignes d'E/S.

En conséquence, les registres **TRIS** se programment avec les valeurs:

TRISA ← F

TRISB ← 00



« 0 » = sortie

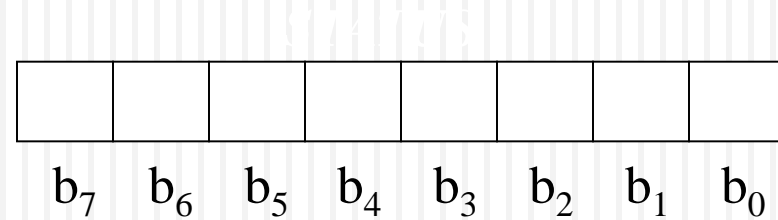
« 1 » = entrée

Register of *STATUS*

Le registre d'état *STATUS* est un registre *8 bits*.

Le rôle de ce registre est de donner diverses informations à l'utilisateur sur *l'état* de fonctionnement ou sur le résultat d'une opération.

On s'intéressera en fait à seulement 2 bits du registre d'état.



*Le bit b_2 : **Z** (Zéro)*

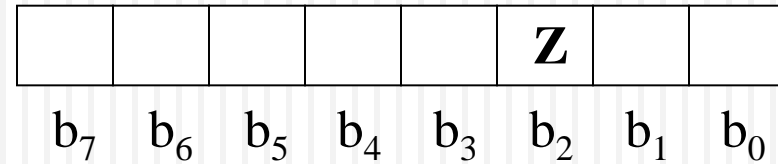
Lorsqu'une opération arithmétique ou logique est réalisée, le bit **Z** est mis à **1** si le résultat est nul et à **0** dans le cas contraire.

Ex1: L'unité centrale effectue l'opération $7-6=1$.

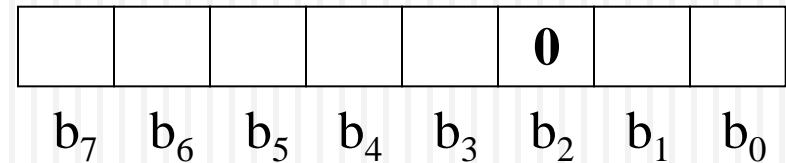
$Z=0$

Ex2: L'unité centrale effectue l'opération $7-7=0$.

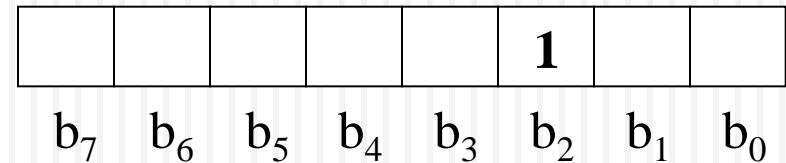
$Z=1$



Ex1:



Ex2:



Register of STATUS

Le bit b_0 : **C** (Carry = retenue)

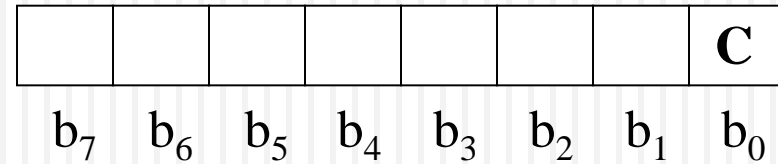
Ce bit est positionné à **1** si une addition ou une soustraction génère *une retenue* depuis le bit de poids fort.

Ex1: L 'unité centrale effectue 1 'opération sur 8 bits:

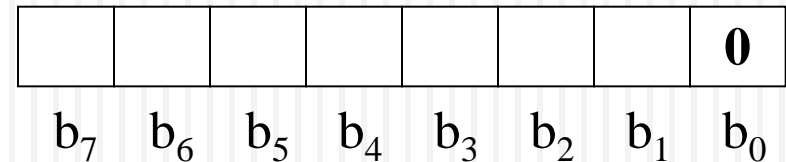
FE+01=FF
C=0

Ex2: L 'unité centrale effectue 1 'opération hexadécimale:

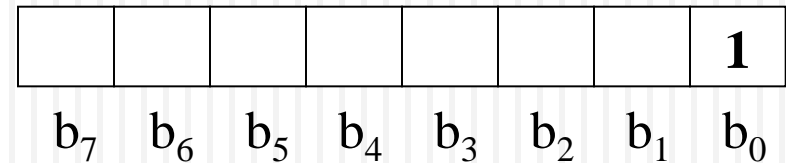
FF+1=00 (et une retenue)
C=1



Ex1:



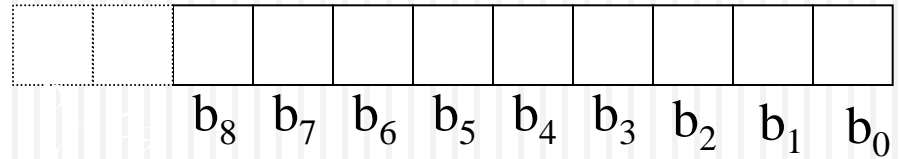
Ex2:



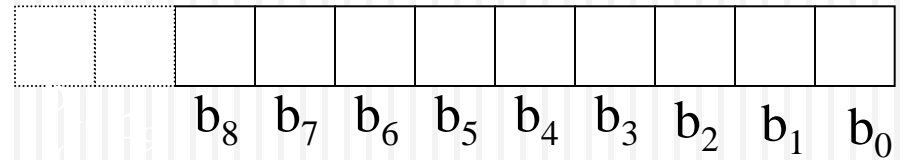
Compte programme (PC)

Le registre **PC** est un registre spécifique **9 ou 11 bits**, suivant le modèle de PIC.

C 'est en fait un **compteur ordinal** qui contient l '**adresse** en mémoire de la **prochaine instruction** à exécuter.



Prenons l'exemple d'un *programme* objet stocké en *mémoire de programme* à partir de l'adresse *000*:



Adresses	Prog.
000	0100
001	30FF
002	3E01
003	0085
004	...
005	...

Compte programme PC

1^{er} cycle machine:

Le registre PC est chargé avec l'*'adresse* de la première instruction du programme.

		0	0	0	0	0	0	0	0	0
		b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀

Adresses	Prog.
000	0100
001	30FF
002	3E01
003	0085
004	...
005	...

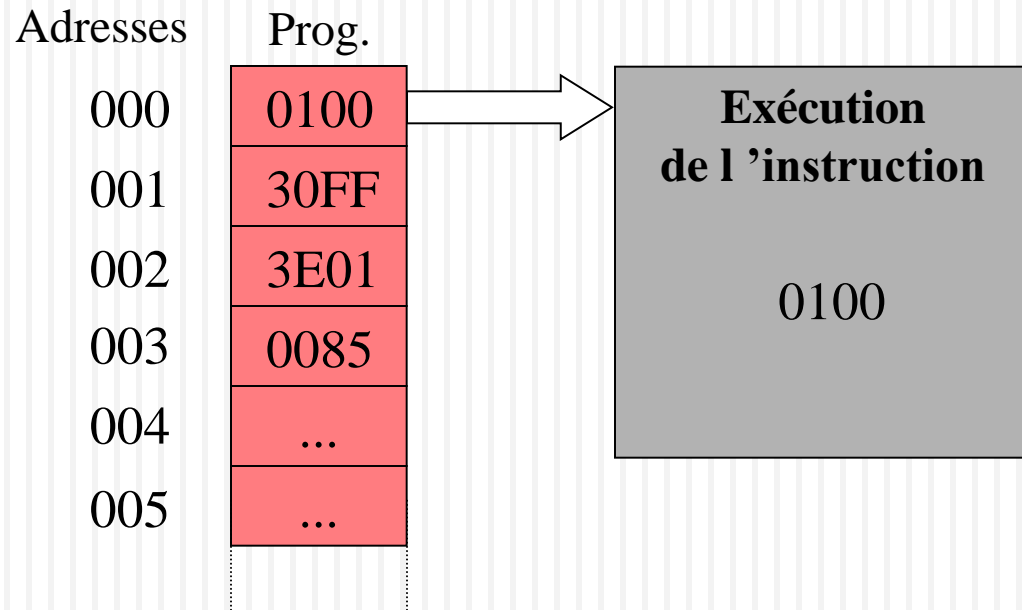
Compte programme PC

2^{ème} cycle machine :

De façon simultanée,

- Le registre PC s' **incrémente**.
- La donnée précédemment pointée par le registre PC est **exécutée**.

		0	0	0	0	0	0	0	0	0
		b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀



Compte programme 70

3^{ème} cycle machine :

De façon simultanée,

- Le registre PC s' *incrémente*.
- La donnée précédemment pointée par le registre PC est *exécutée*.

		0	0	0	0	0	0	0	0
		b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁ b ₀

Adresses

Prog.

000

0100

001

30FF

002

3E01

003

0085

004

...

005

...

**Exécution
de l' instruction**

30FF

Compte programme PC

4^{ème} cycle machine :

De façon simultanée,

- Le registre PC s' **incrmente**.
- La donnée précédemment pointée par le registre PC est **exécutée**.

		0	0	0	0	0	0	0	1	0
		b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀

Adresses

Prog.

000

0100

001

30FF

002

3E01

003

0085

004

...

005

...

**Exécution
de l' instruction**

3E01

Compte programme PC

5^{ème} cycle machine :

De façon simultanée,

- Le registre PC s' **incrémente**.
- La donnée précédemment pointée par le registre PC est **exécutée**.

		0	0	0	0	0	0	0	0	0
		b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀

Adresses

Prog.

000

0100

001

30FF

002

3E01

003

0085

004

...

005

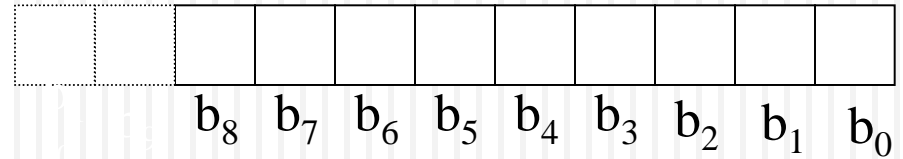
...

**Exécution
de l' instruction**

0085

Compte-programme PC

PC



Conclusion

Le registre PC contient donc à l'instant t l'*adresse* de la *prochaine instruction* à exécuter.

Ainsi, lorsqu'une instruction est exécutée, la suivante est *déjà pointée* par le registre PC.

Adresses	Prog.
000	0100
001	30FF
002	3E01
003	0085
004	...
005	...

**Exécution
de l'instruction**

La base de temps

Comme tous les circuits microprogrammés, les microcontrôleurs **PIC 16Cxx** fonctionnent à partir d'une base de temps (horloge) appliquée par des composants externes.

Ainsi, les PIC peuvent adopter 4 modèles d'horloge qui sont:

- Version **XT**
- Version **HS**
- Version **RC**
- Version **LP**

Version XT:

oscillateur à quartz jusqu'à 4 MHz.

Version HS (High Speed):

oscillateur à quartz jusqu'à 20 MHz.

Version RC (Résistance-Condensateur):

oscillateur RC jusqu'à 4 MHz.

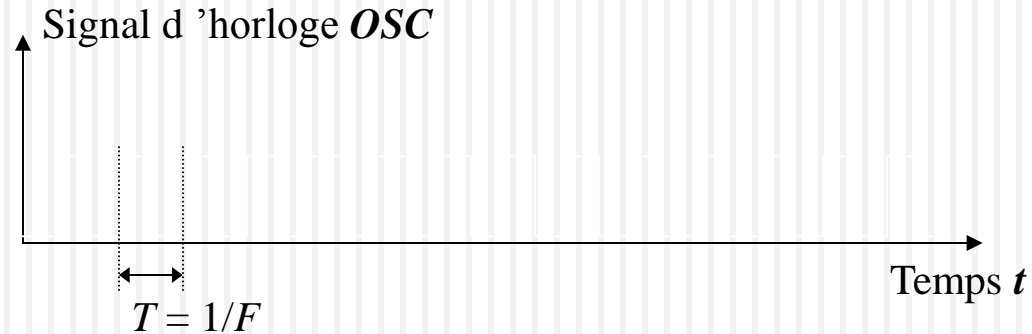
Version LP (Low Power):

oscillateur à quartz jusqu'à 200 kHz.
Prévu pour des applications à faible consommation.

Le rôle de l'horloge est de *cadencer* les différentes opérations effectuées par le microcontrôleur et notamment l'*exécution des instructions* du programme.

Ainsi, le signal d'horloge possède les caractéristiques suivantes:

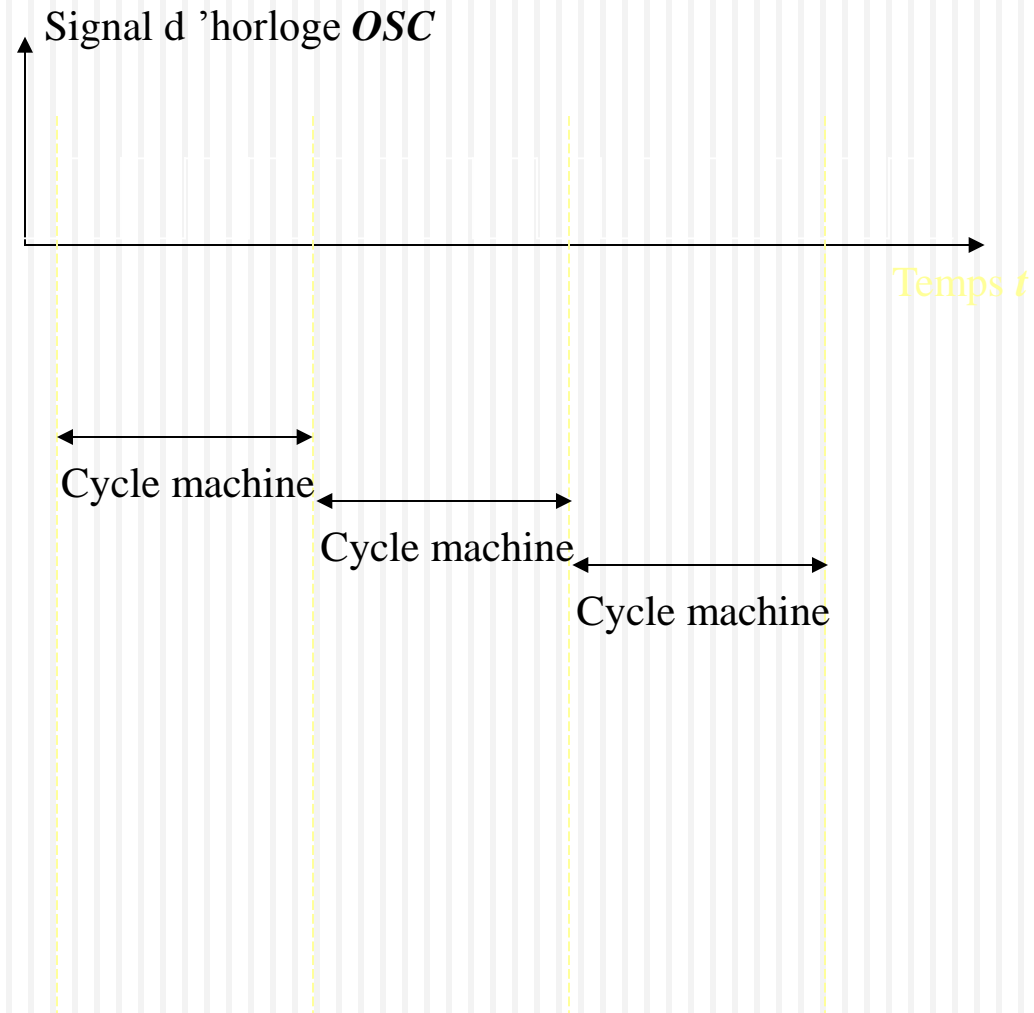
- Signal *carré*.
- De fréquence F et de période T .



Le signal d 'horloge ***OSC*** est en fait délivré par un oscillateur *externe* qui peut être un *quartz* ou une cellule ***RC***.

Ce signal appliqué au PIC est ensuite, de façon interne, *divisé par 4*.

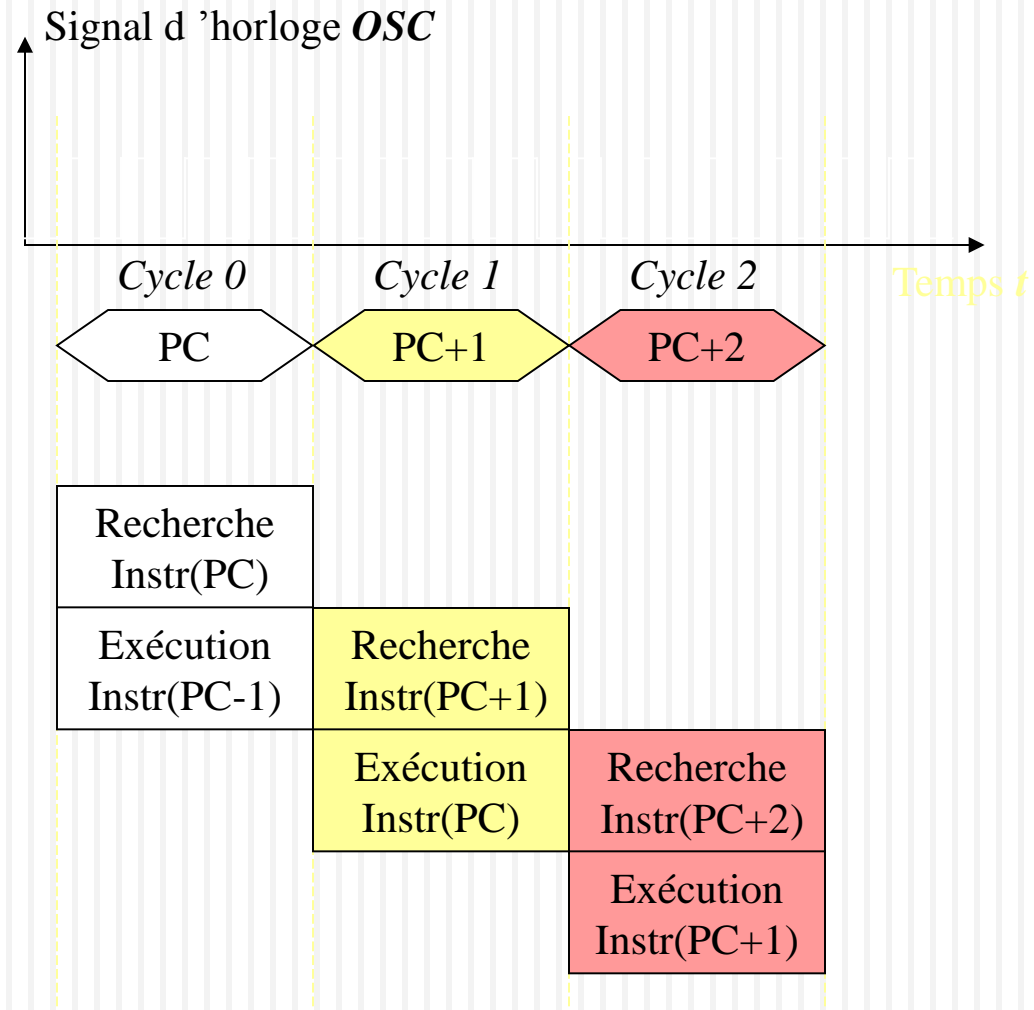
On appelle alors *cycle machine* la durée caractérisant 4 périodes d 'horloge.



Cette base de temps permet alors de rythmer l'exécution des instructions du programme:

Ainsi, au cours de chaque cycle machine:

- **Incrémentation** du registre **PC**.
- **Recherche** de l'instruction dont l'adresse est contenue dans le registre PC.
- **Exécution** de l'instruction qui était pointée par le registre PC au cours du *cycle précédent*.



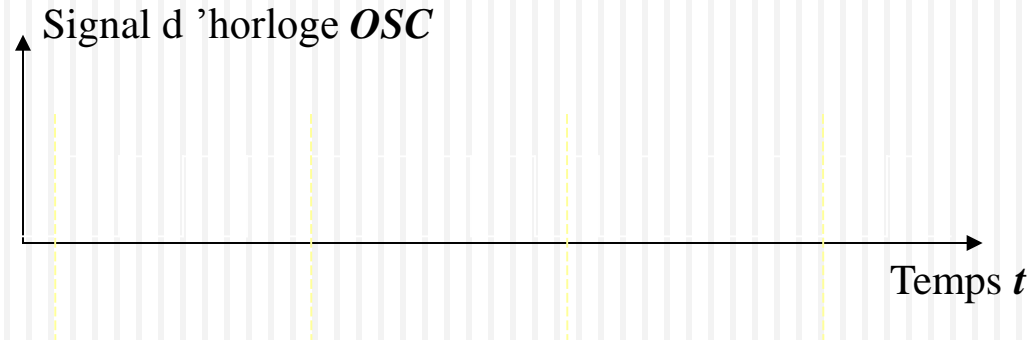
Conclusion:

Il faut donc *un cycle machine* pour exécuter une instruction, soit *4 périodes d'horloge*.

Ex:

Dans le cas d'un oscillateur à quartz à **20MHz**, le temps d'exécution d'une instruction est donc:

200 ns



$$\text{Instruction cycle time} = 4 \times T_{\text{osc}}$$

$$\text{Si } F_{\text{osc}} = 20 \text{ MHz} \quad T_{\text{osc}} = 1/F_{\text{osc}}$$

$$T_{\text{osc}} = 50 \text{ ns}$$

$$\text{D'où } 4 \times T_{\text{osc}} = 200 \text{ ns}$$

Le jeu d' instructions et les modes d' adressage

Outre la facilité de mise en œuvre matérielle, l' intérêt des microcontrôleurs PIC réside dans le *jeu d' instruction* et les *modes d' adressage* considérablement réduits par rapport à d' autres structures programmables.

Jeu d'instructions

En plus de bénéficier d'une architecture dite Harvard, les microcontrôleurs **PIC** sont constitués autour d'une architecture appelée **RISC**.

Ainsi, contrairement à de nombreux circuits mettant en jeu une centaine d'instructions différentes, les **PIC** voient leur nombre d'instructions limités à **33 ou 35**.

Région **I**nstruction **S**imple **C**ircuit
= *Circuit à jeu d'instructions réduit*

33 ou 35 instructions seulement!

Les différents *mnémoniques* du jeu d'instructions adoptent certaines appellations dont il est nécessaire d'être informé:

- *k* est un littéral, c'est-à-dire une valeur codée sur un octet (8 bits).

Ex:

Le littéral *k* (1 octet) est placé dans le registre de travail W.

$$k \rightarrow W$$

Les différents *mnémoniques* du jeu d'instructions adoptent certaines appellations dont il est nécessaire d'être informé:

- *k* est un littéral, c'est-à-dire une valeur codée sur un octet (8 bits).
- *f* est le symbole correspondant à un registre.

Ex:

Le contenu du registre de travail *W* est transféré dans le registre représenté par *f*.

$$W \rightarrow f$$

Les différents *mnémoniques* du jeu d'instructions adoptent certaines appellations dont il est nécessaire d'être informé:

- *k* est un littéral, c'est-à-dire une valeur codée sur un octet (8 bits).
- *f* est le symbole correspondant à un registre.
- *b* est le numéro du bit concerné par l'instruction.

Ex:

303 43

Le bit *b* du registre *f* est mis à zéro (Bit Clear).

$0 \rightarrow \text{bit } b \text{ de } f$

Les différents mnémoniques du jeu d'instructions adoptent certaines appellations dont il est nécessaire d'être informé:

- k est un littéral, c'est-à-dire une valeur codée sur un octet (8 bits).
- f est le symbole correspondant à un registre.
- b est le numéro du bit concerné par l'instruction.
- d caractérise le registre où doit être placé le résultat de l'opération.

Ex:

Le contenu du registre W est ajouté au contenu du registre f .

Si $d=0$ le résultat est placé dans W .

Si $d=1$ le résultat est placé dans f .

Si $d=0$ $W+f \rightarrow W$

Si $d=1$ $W+f \rightarrow f$

Modes d'adressage

La encore, les *modes d'adressage* sont réduits puisqu'on en compte que 4:

- Adressage immédiat.

Ex: *NOV 17 1983*

La donnée manipulée (k) est codée *immédiatement* avec l'instruction.

Modes d'adressage

La encore, les *modes d'adressage* sont réduits puisqu'on en compte que 4:

- Adressage immédiat.
- Adressage direct.

Ex:

Le registre concerné (*f*) est codé *mmmm* dans l'instruction.

Adressage

La encore, les *modes d'adressage* sont réduits puisqu'on en compte que 4:

- Adressage immédiat.
- Adressage direct.
- Adressage bit à bit.

Ex:

303 72

Il permet de manipuler n'importe quel *bit individuel* de n'importe quel registre.

La encore, les *modes d'adressage* sont réduits puisqu'on en compte que 4:

- Adressage immédiat.
- Adressage direct.
- Adressage bit à bit.
- Adressage indirect.

Le registre concerné est atteint via un registre d'indirection. Ce mode est en fait très peu utilisé.