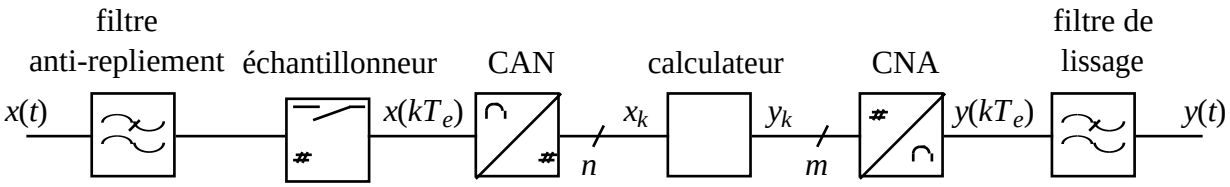
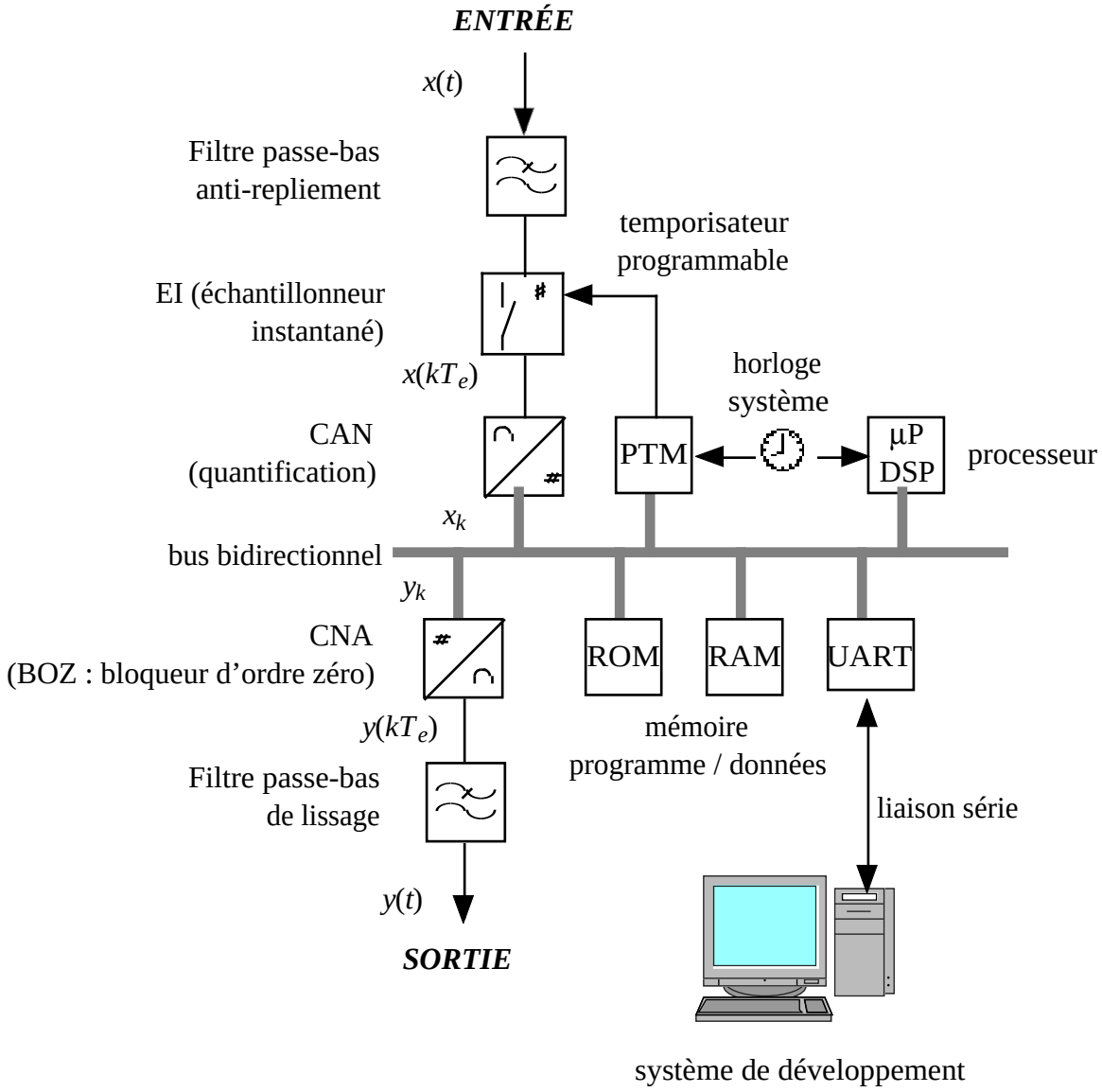
B17 - Chaîne d'acquisition et de traitement numérique

La structure d'une chaîne d'acquisition et de traitement numérique est la suivante :



Détail du système :



Parmi les processeurs existants, on distingue, du plus élémentaire au plus sophistiqué :

UAL : unité arithmétique et logique *(ALU : Arithmetic and Logic Unit)* : ensemble de portes logiques programmables capables de réaliser des opérations booléennes et mathématiques. Souvent réalisée sous forme de "réseau logique programmable" *(PLA : Programmable Logic Array).*

µP : microprocesseur = UAL + séquenceur + registres + compteurs. Permet d'exécuter des programmes complexes.

µC : microcontrôleur = µP + RAM + ROM + PTM + UART + CAN + CNA. Intègre en un seul

boîtier intégré l'ensemble des fonctions contenues sur une carte à microprocesseur.

DSP : microcontrôleur spécialisé pour le traitement numérique des signaux *(DSP : Digital Signal Processing)*

• ***Algorithme de calcul***

- commander le multiplexeur d'entrée et éventuellement l'amplificateur d'instrumentation
- échantillonner et bloquer le signal
- lancer la conversion
- attendre la fin de la conversion
- lire le résultat produit par le CAN
- effectuer les calculs
- envoyer le résultat sur le CNA

********** COMPLEMENTS : Logique microprogrammée **********

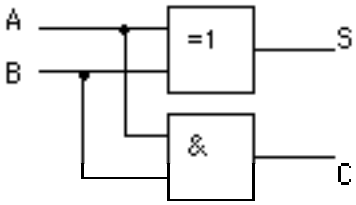
• Exemple (très simplifié !) d'UAL 1 bit

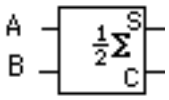
- Semi-additionneur 1 bit

Effectue la somme A + B (2 nombres de 1 bit). Le résultat est sur 2 bits : somme S et retenue C *(Carry)*

```
      A
  +   B
  ---------
=  C   S
```

Décimal			Binaire			
A	B	+	A	B	C	S
0	0	0	0	0	0	0
0	1	1	0	1	0	1
1	0	1	1	0	0	1
1	1	2	1	1	1	0





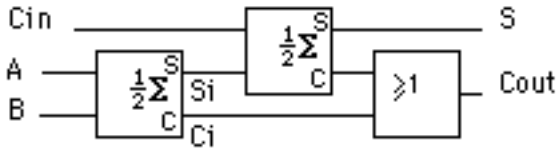
- Additionneur complet

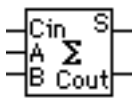
Effectue la somme A + B + C_{in} (C_{in} : retenue d'entrée). Le résultat est sur 2 bits : S et retenue C_{out} . Cela permet de "cascader" des additionneurs 1 bit pour ajouter des nombres de taille quelconque.

```
      A          0          0          1          1
  +   B      +   0      +   1      +   0      +   1
  ---------  ---------  ---------  ---------  ---------
   Ci  Si      0  0       0  1       0  1       1  0
  +   Cin    +   Cin    +   Cin    +   Cin    +   Cin
  ---------  ---------  ---------  ---------  ---------
  Cout  S      0  Cin     0  1       0  1       1  Cin
                        ou 1  0    ou 1  0
```

Décimal				Binaire				
Cin	A	B	+	Cin	A	B	Co	S
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	1
0	1	0	1	0	1	0	0	1
0	1	1	2	0	1	1	1	0
1	0	0	1	1	0	0	0	1
1	0	1	2	1	0	1	1	0
1	1	0	2	1	1	0	1	0
1	1	1	3	1	1	1	1	1

La retenue finale Cout vaut 1 si Ci vaut 1 OU si la retenue de Si+Cin vaut 1.





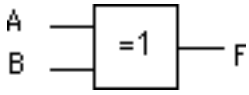
- Le OU Exclusif, plus petit ordinateur du monde !

La table de vérité du OU Exclusif peut être interprétée comme une fonction OUI/NON programmable (avec par ex A : entrée de programmation ; B : entrée de donnée).

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

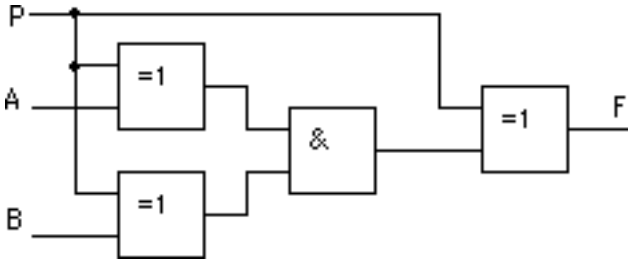
⇔

A	F
0	B
1	B&



- Unité logique programmable 1 bit

Elle est basée sur l'utilisation des règles de De Morgan.

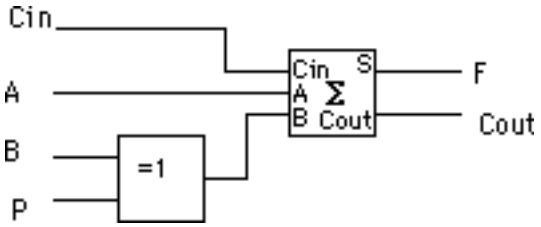


P	fonction
0	F = A ET B
1	F = A OU B

- Unité arithmétique programmable 1 bit

NB : en code complément à 2, la quantité – B est codée par le nombre binaire B + 1. La soustraction A – B est donc effectuée en faisant l'addition A + B + 1. C_{in} et C_{out} (ou leurs négations logiques pour la soustraction) sont respectivement les retenues d'entrée et de sortie de l'additionneur (noté $\sum$).

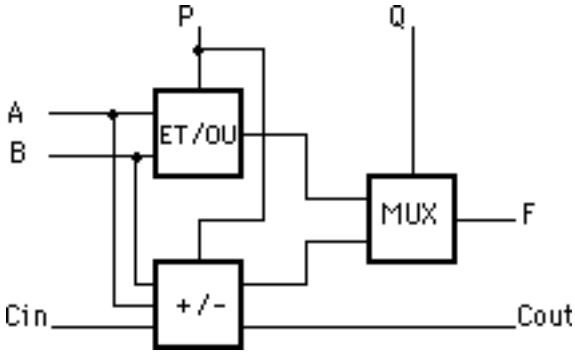
P	Cin	F	Cout
0	0	A + B	Cout
0	1	A + B + 1	Cout
1	0	A - B - 1	C&o&u&t&
1	1	A - B	C&o&u&t&

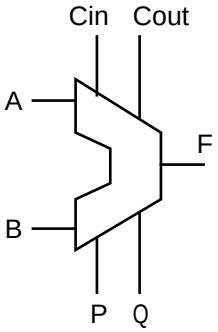


- Unité arithmétique et logique 1 bit

Un multiplexeur permet de choisir entre opérations logiques ou opérations arithmétiques.

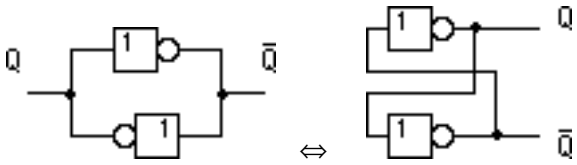
Q	P	F	Cout
0	0	A OU B	X
0	1	A ET B	X
1	0	A + B + Cin	Cout
1	1	A – B – C&i&n&	C&o&u&t&





• *Logique séquentielle (rappels)*

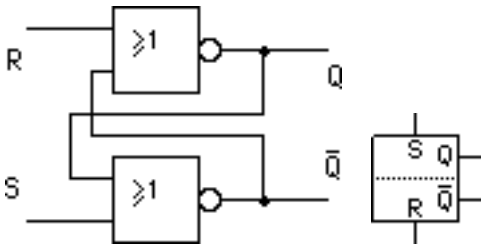
- cellule mémoire à lecture seule avec sorties complémentées



NB : ce système reste dans l'état où il est indéfiniment !

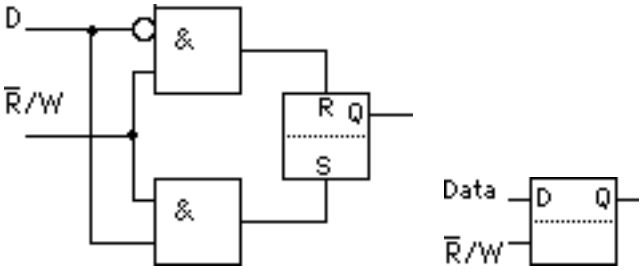
- verrou RS (RS latch) : cellule mémoire à lecture / écriture (2 lignes d'écriture, R et S)

R	S	Q(n+1)	
0	0	Q(n)	mémoire
0	1	1	RA1 (Set)
1	0	0	RAZ (Reset)
1	1	Q = Q& = 0	interdit !



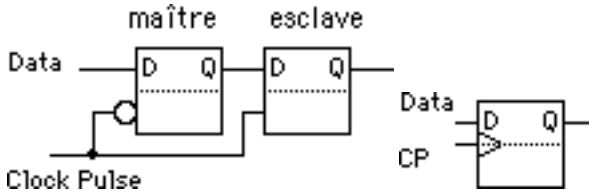
- verrou type "D" (D latch) : cellule mémoire à lecture / écriture (1 seule ligne d'écriture, D)

R&/W	D	Q(n+1)	
0	X	Q(n)	lecture
1	0	0	écriture
1	1	1	écriture

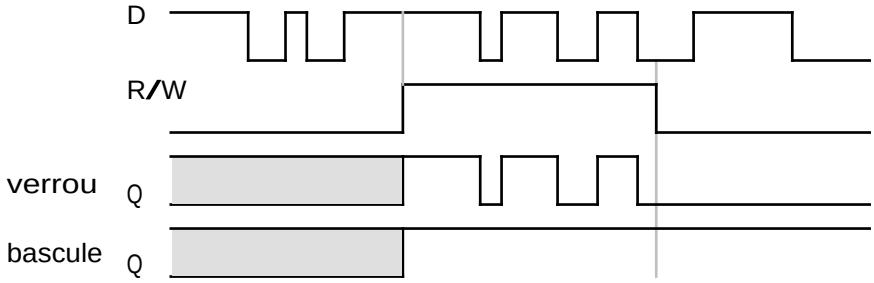


- bascule maître-esclave type "D" (D flip-flop) : cellule mémoire à lecture / écriture sur front montant

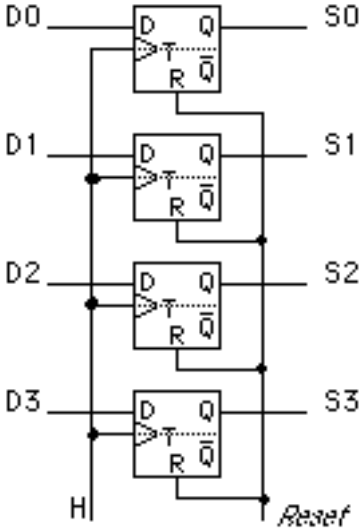
CP	D	Q(n+1)	
X	X	Q(n)	lecture
☾	0	0	écriture
☾	1	1	écriture

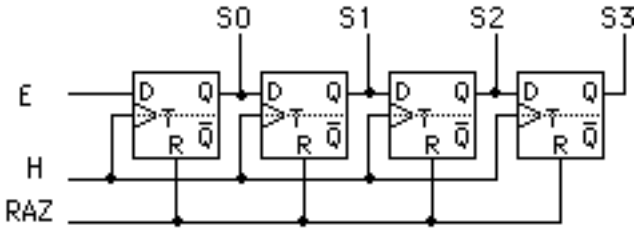


chronogrammes des verrou et bascule "D" :



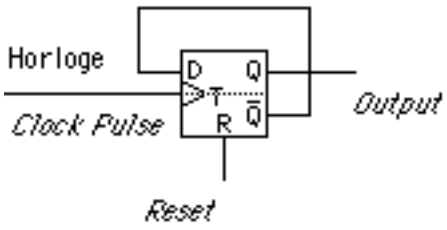
- registres : chargement parallèle, sortie parallèle *entrée série, sortie parallèle (S1...S3) ou série (S3)*

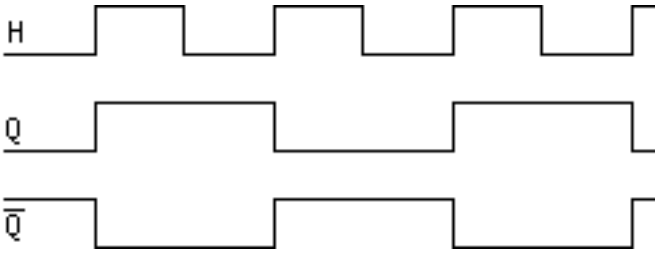




	S0	S1	S2	S3
Reset	0	0	0	0
H (Î	D0	0	0	0
H (Î(Î	D1	D0	0	0
H (Î(Î(Î	D2	D1	D0	0
H (Î(Î(Î(Î	D3	D2	D1	D0

- compteur / diviseur de fréquence par deux

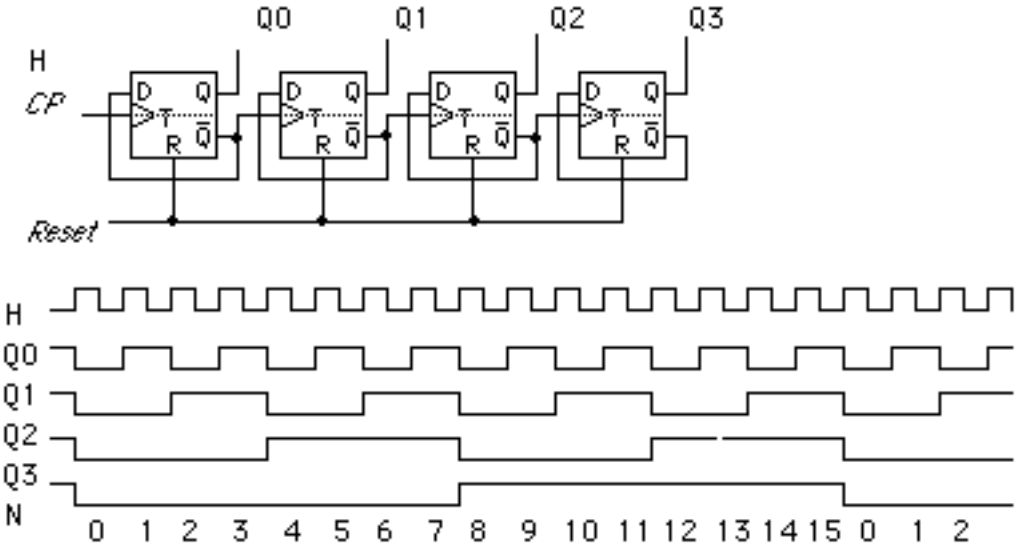




- *compteur*

NB : les sorties inverseuses (Q&) permettent d'obtenir un décompteur.

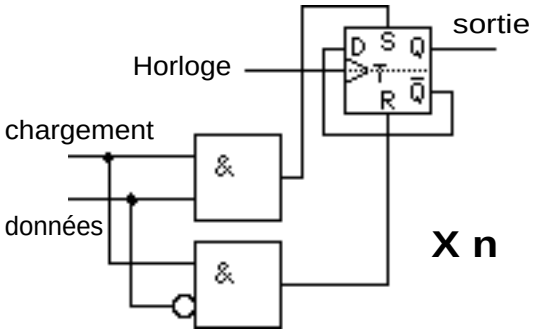
Q3	Q2	Q1	Q0	N
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15



- *compteur prépositionnable* (presettable counter)

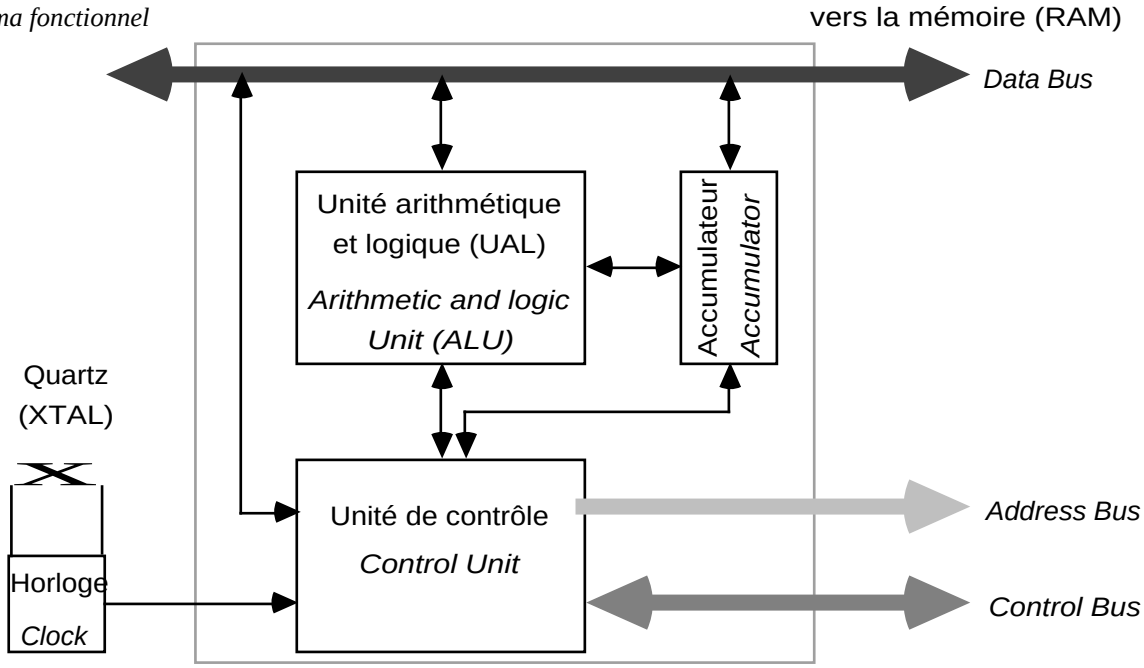
Commence le comptage à partir d'un nombre quelconque compris entre 0 et Nmax préalablement chargé par l'intermédiaire des entrées R et S.

chargement *PRESET*	données *data*	sortie *output*
0	X	Q
0	X	Q
1	0	0
1	1	1



• ***Exemple (très simplifié !) de microprocesseur***

- *schéma fonctionnel*



Un microprocesseur est composé d'une UAL, d'un registre de travail et d'une unité de contrôle, et est associé à une

mémoire vivre (RAM) externe.

Le registre de travail ou "accumulateur" est un registre parallèle constituant une sorte de "carnet de brouillon" où l'UAL lit et écrit les données et les résultats des opérations qu'elle effectue. L'unité de contrôle est un séquenceur (en ROM) qui exécute à l'intérieur du microprocesseur les microactions nécessaires pour accomplir la suite des opérations élémentaires constituant une instruction de programme.

La RAM contient les octets d'instructions et de données : chaque octet est spécifié par un numéro (son "adresse") activé par un "décodeur d'adresse".

Un "bus" est un ensemble de lignes électriques parallèles. Les données transitent par un "bus de données" bidirectionnel *(data bus)* ; le microprocesseur spécifie les adresses à la mémoire vive par l'intermédiaire d'un "bus d'adresse" unidirectionnel *(address bus)* ; l'ensemble des lignes de contrôle est appelé "bus de contrôle" *(control bus)*.

Le tout est piloté par une horloge externe.

- *schéma électrique (très) simplifié.* ***(Voir page suivante)***

Statistiquement, 60% d'un programme consiste à effectuer des mouvements de données (ou adressage) entre le microprocesseur et la mémoire. Avant d'être un calculateur, le microproceseur est surtout un facteur !

Typiquement, une instruction de programme comprend les étapes suivantes, chaque étape correspondant à 1 cycle d'horloge (ou "cycle machine") :

1) Lire en mémoire l'octet codant l'instruction ; le charger dans le compteur prépositionnable (ou registre d'instruction) du séquenceur ; lancer le microprogramme correspondant (étapes 2 à 5).
2) Lire en mémoire les données et transférer celles-ci dans les registres du µP (dont l'accumulateur).
3) Programmer l'UAL pour effectuer les opérations demandées ; stocker le résultat dans l'accumulateur.
4) Sauvegarder le résultat en le transférant de l'accumulateur vers la mémoire.
5) Incrémenter le compteur programme pour pointer l'adresse de la prochaine instruction.

NB :

- le registre d'état *(status register)* est un registre particulier où l'UAL sauvegarde certaines caractéristiques propres à l'opération qui vient d'être effectuée, sous forme de bits d'état : retenue *(C : Carry)* ; dépassement *(V : overflow)* ; résultat nul *(Z)* ; résultat négatif *(N)* ; etc. Ces indicateurs sont utilisés par les instructions de branchement, notamment dans les tests *(IF... THEN....)*
- le registre d'adresse *(address register)* sert à la manipulation et au calcul d'adresses, notamment pour les instructions opérant sur des tableaux (déplacement, tri,...).

- *espace mémoire ou espace d'adressage*

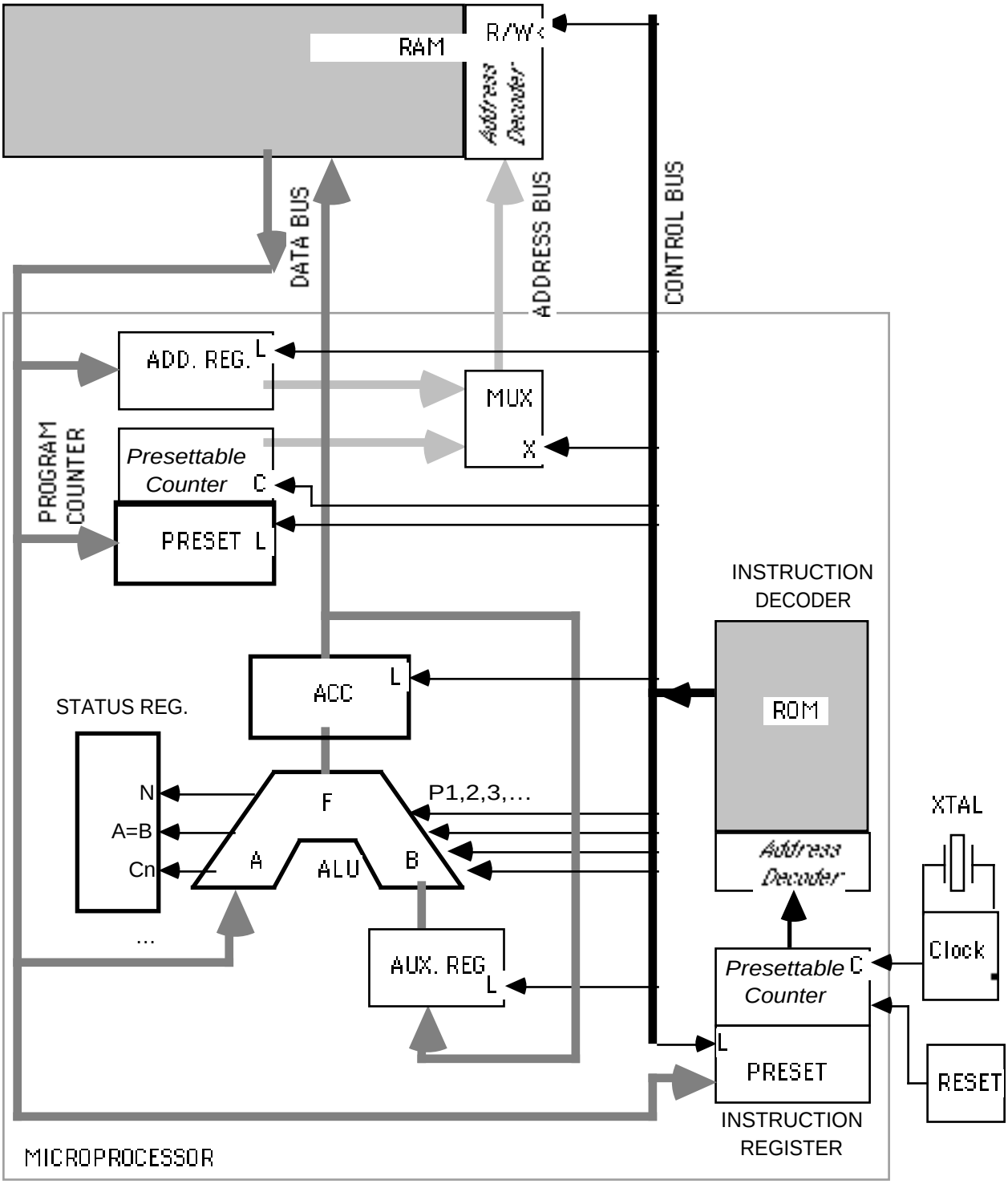
Les octets représentant les instructions ou les données sont stockés les uns à la suite des autres dans l'espace mémoire, qui va de l'adresse la plus basse à l'adresse la plus haute. Par exemple, avec 16 bits d'adresse, l'espace mémoire commence à l'adresse $0000 jusqu'à l'adresse $FFFF, soit $2^{16} - 1 = 65535$ en décimale (on dit : 64 kilooctets).

Certaines adresses désignent des octets sauvegardés en RAM, d'autres en ROM. D'autres adresses sont celles de dispositifs d'entrée/sortie : CAN, CNA, UART, etc. Enfin, certaines adresses peuvent être inutilisées.

Dans un système numérique, l'organisation de l'espace d'adressage est indiquée sous forme d'une "carte mémoire" *(memory map)* qui décrit cette organisation (à quelles adresses se trouvent la RAM, la ROM, les entrées/sorties, etc). Pour l'utilisateur, cette information est essentielle !

- *programmation*

Une instruction se compose d'un code opératoire *(Op. Code)* et d'un opérande (une donnée ou une adresse). Il existe principalement trois types de données, selon la méthode (appelée "mode d'adressage") par laquelle le microprocesseur va chercher la donnée en mémoire :

type *constante*	opérande = donnée immédiate, insérée dans le programme lui-même		
	BASIC :	x = x + 8	
	µP :	ADDA #$08	adressage *immédiat* : ajoute à l'accumulateur la quantité 8 située immédiatement après le code opératoire de l'instruction ADD
type *variable*	opérande = adresse de la donnée		
	BASIC :	x = x + y	
	µP :	ADDA $0100	adressage *étendu* : ajoute à l'accumulateur la quantité située à l'adresse $0100
types *pointeur*	opérande = pointeur + offset		
(n'existe pas en BASIC)		ADDA 5,X	adressage *indexé* : ajoute à l'accumulateur la quantité située à l'adresse X + 5, où X est le contenu d'un registre d'adresse, et 5 le déplacement par rapport à X.
ou	opérande = adresse de l'adresse de la donnée		

ADDA [$0100] adressage *étendu indirect :* ajoute à l'accumulateur la quantité située à l'adresse stockée à l'adresse $0100.