MICROCONTRÔLEUR (µC)

- **3 TYPES DE SIGNAUX :**

- **Analogique** = grandeurs continues (défini partout // ex. : température)
- **Logique** (ex. : bouton poussoir)
- **Numérique** = grandeurs discrètes (nombre fini de valeurs // défini sur une plage de valeurs)

Remarque : Signal logique => signal numérique

Cas particulier: signal échantillonné : signal discret (=numérique) dont les valeurs sont prises sur un signal continu (=analogique).

CAN : Convertisseur Analogique/Numérique
CNA : Convertisseur Numérique/Analogique

- **DEFINITIONS :**

Ordinateur = machine électronique capable d'exécuter des instructions effectuant des opérations sur des nombres (calculette). Il est donc multitâche.

contient →
- **Processeur** = Exécute les instructions <=> exécute le programme contenu dans la mémoire vive.
- **Mémoire vive** (Mémoire de données) = RAM : support stockant les données (variables intermédiaires // perte des données à la mise hors tension).
- **Disque dur** (Mémoire morte || Mémoire programme) = ROM : stock les programmes.

Périphérique (entrées/sorties) : stock les données secondaires permettant d'échanger des infos avec l'extérieur.
Processus = programme en cours d'exécution
Logiciel = ensemble de fichier/programmes
Programme = suite d'opérations + données
Compilateur ~ Traducteur (en informatique)

MICROCONTROLEUR = ordi qui exécute toujours la même séquence : il est donc monotâche. [le processeur est un microprocesseur µP]

Conclusion : Ordi : multitâche / µC : monotâche

- **2 TYPES DE FICHIERS :**

→ fichiers **ressources** : images /vidéos/musique …
→ fichiers **programmes** : opérations pour être exécutées

- **2 TYPES DE PROGRAMMES :**

→ **Binaire** [directement compréhensible par le processeur] = ensemble d'instructions exécutables par le processeur : « Ce que l'ordinateur comprend ».
→ **Source** [à besoin d'être compilé] = ensemble d'opérations abstraites décrivant des actions à effectuer : « Ce que l'on comprend ».

Ex. : Déclare x=0 ; Ajoute 42 à x -----compilation------> 0x4883C02a

- **ECRITURE & COMPILATION :**

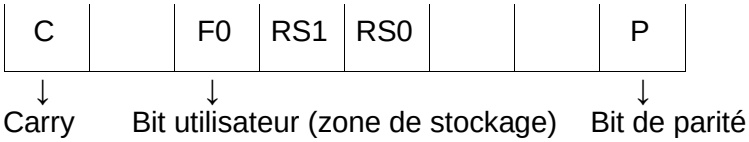
1) Écriture des programmes effectuée en langage assembleur.
2) Assembler/compiler permet de traduire en langage compréhensible pour l'ordinateur.
3) Un programmateur transfert le programme compilé dans la mémoire du µC.

- **MODES D'ADRESSAGES :**

immédiat : valeur numérique (ex. : #30h)
direct : adresse (ex. : 30h)
indirect : le contenu du registre R0 ou R1 (ex. : @R0)
registre : A ; R0 à R7 ; DPTR ; C

Remarque :

- le PSW du µC = Program Status Word



- le PC permet de suivre le déroulement d'un programme = le compteur.

4 banques existent (0 à 3) :

Banque 0	RS0 = 0	RS1 = 0
Banque 1	RS0 = 1	RS1 = 0
Banque 2	RS0 = 0	RS1 = 1
Banque 3	RS0 = 1	RS1 = 1

Par défaut on se situe dans la banque 0.

- **CORRESPONDANCE pseudo-code/assembleur : (Exemples)**

(30h) / 30h (direct)
30h / #30h (immédiat)
((R0)) / @R0 (indirect)
(A) / A (registre)

- **STRUCTURE DES INSTRUCTIONS :**

en assembleur : MNEMONIQUE + OPERANDE

ex : MOV A, 30h
A (destinataire) et 30h (source) sont les opérandes
MOV est la mnémonique.

BYTE = nombre d'octets nécessaires au codage de l'instruction

CYCLE = nombre de cycles machines nécessaires à l'exécution de l'instruction
cycle = C = 12 / (fréquence du µC)

INSTRUCTIONS DE SAUT
LJMP = saut avec adresse de 16 bits
AJMP = saut avec adresse de 11 bits
NOP = ne fait rien – temporisation

INSTRUCTIONS ARITHMETIQUES
ADD = ajout
INC = incrémentation
DEC = décrémentation
SUBB = soustraction
MUL = multiplication
DIV = division

INSTRUCTIONS LOGIQUES
ANL = « ET »
ORL = « OU »
XRL = « OU EXCLUSIF » ⊕
CLR = mise à 0
CPL = complémentation

INSTRUCTION BOOLEENES (sur bits)
CLR = mise à 0
SETB = mise à 1
CPL = complémentation
ANL = « ET »
ORL = « OU »
MOV = déplacement

INSTRUCTIONS DE TRANSFERT
MOV = écrase l'ancienne valeur du destinataire avec celle de la source

- **STRUCTURE D'UN PROGRAMME (en assembleur)**

```
; Nom_du_programme (tout ce qui suit « ; » est un commentaire !!!)

; Déclaration des variables

Var1    equ     valeur ; equ = équivalant, Var1 est une valeur/adresse
Var2    bit     adresse ; Var2 est une adresse bit

; Implantation des adresses

        org     0000h
        LJMP    0030h
        org     0030h

; Programme principal
début:

 fin :    LJMP début      ; on fait boucler le µC
          end             ; on ignore ce qu'il y a après
```