Itérations

DBEQ

Décrémente un registre et branche si le registre vaut 0

DBNE

Décrémente un registre et branche si le registre ne vaut pas 0

IBEQ

Incrémente un registre et branche si le registre vaut 0

IBNE

Incrémente un registre et branche si le registre ne vaut pas 0

TBEQ (modifie pas le registre)

Test le registre et branche si le registre vaut 0

TBNE (modifie pas le registre)

Test le registre et branche si le registre ne vaut pas 0

- Ces instructions ont 2 opérandes: le registre et le label (offset)
- l'offset doit être compris dans [-256 ; 255] depuis le début de la prochaine instruction

```
; Début de l'itération
        ldaa #7         ; A = 7
loop:   incb            ; Incrémente B de 1
        dbne A,loop     ; Décrémente A de 1 et saut à loop si A !=0
; Fin de l'itération
```

Boucle infinie

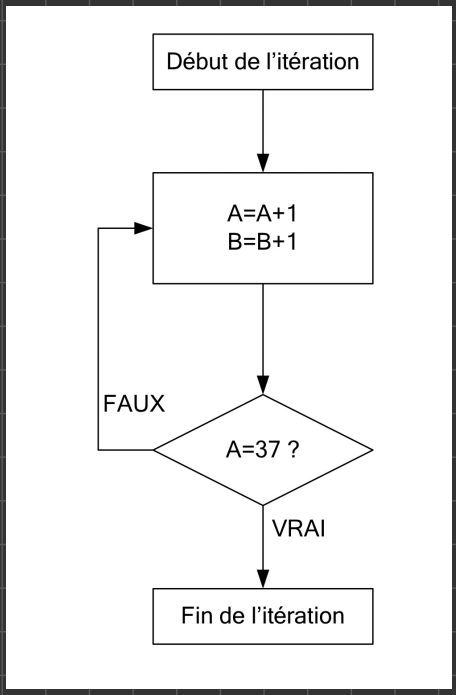
les programmes pour microcontrôleur sont toujours infinies (sauf le bloc d'initialisation)

```
; Code d'initialisation
Loop:
   ; Action à répéter
   bra Loop ; Utilisez LBRA si offset plus grand que 8 bits
```

do ... while
(fais qqch tant que la condition est vraie)

Cette itération fonctionne comme ceci:

1) Réalise une action
2) Test une condition
3) Branche à l'action (1) si la condition de fin est vraie

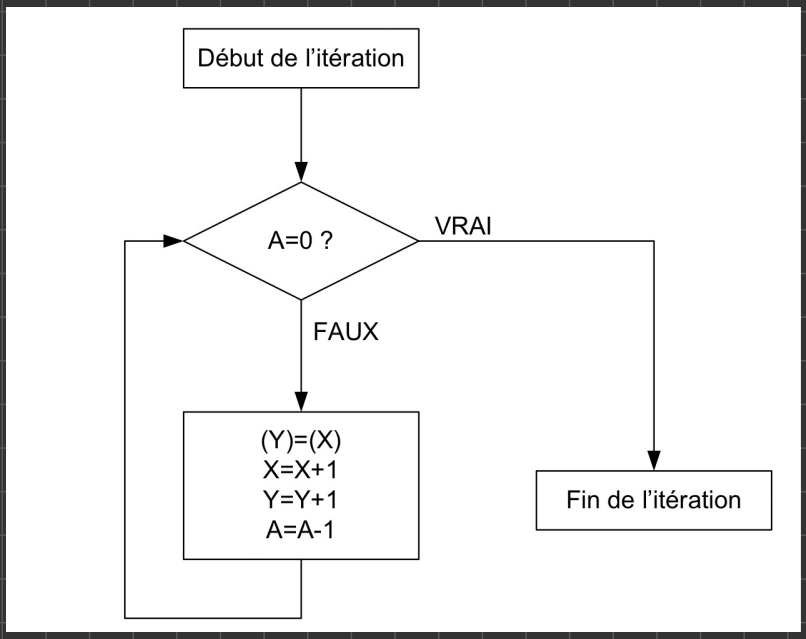


en C

```c
char A, B;
/* ... */
do
   {
      A++;
      B++;
   }while (A != 37);
```

```
; Début de l'itération
loop:   inca          ; Incrémente A
        incb          ; Incrémente B
        cmpa #37      ; A - 37
        bne loop      ; Saut à loop si A != 37
; Fin de l'itération
```

while ...
(la condition est au début)



En C

```c
unsigned char A;
unsigned char *X, *Y;
/* ... */
while (A != 0)
   {
      *X++ = *Y++;
      A--;
   }
```

Assembleur

```
L1:   tsta              ; A - 0
      beq L2            ; Saut à L2 si A =0
      movb 1,X+,1,Y+    ; Transfert le contenu de l'adresse de X vers
                        ; l'adresse de Y. On incrémente X et Y de 1
      deca              ; Décrémente A de 1
      bra L1            ; Saut à L1
L2:
```

Tableaux

- C'est une suite de cases mémoires en RAM
- On accède aux éléments d'un tableau à l'aide de l'indexage

```
unsigned char a1[10];
```

```
;-------------------------------------------------------------
; Section pour les variables et les données --> zone de RAM
;-------------------------------------------------------------
DEFAULT_RAM:      SECTION
a1: ds.b 10                    ; Tableau de 10 bytes
```

Tables

- C'est un tableau mais en ROM

```
const int t1[5] = {3, 5, 10, 153, 210};
```

```
;-------------------------------------------------------------
; Section pour le code et les constantes --> zone de ROM
;-------------------------------------------------------------
DEFAULT_ROM:      SECTION
t1: dc.w 3,5,10,153,210    ; Tables de 5 mots (words)
```

Accéder à un tableau ou une table

Il y'a 2 approches possibles:

- Séquentiel

  On accède tour à tour à chaque éléments

- Aléatoire

  On accède à une seule location mémoire à la fois à l'aide d'un index

Ces 2 approches utilises un registre d'index et des modes d'adressage indexé

Pour le mode séquentiel l'adresse de début du tableau est chargé dans un registre d'index puis on accède à chaque location en utilisant le mode d'adressage indexé post incrémenté

```
      ldx     #a1         ; Charge l'adresse de la 1ère case du tableau dans X
l1:   movb    #7,1,x+     ; 7 à l'adresse contenue dans X puis X=X+1
      cpx     #a1+10      ; Test si on est à la fin du tableau
      bne     l1          ; On continue si on n'est pas au bout du tableau
```

L'adresse du $N^{ième}$ élément d'un tableau de bytes de taille S sera: $A + (N * S)$, avec A qui est l'adresse du 1^{er} élément

Si on prend l'exemple de

Data2 = Tab[Index]

avec Index une variable de 8 bits non-signée, Data2 une variable de 16 bits signée et Tab un tableau de 16 bits.

On doit donc incrémenter Index de 2 pour passer d'un index à l'autre

```
ldx     #Tab        ; Adresse de la 1ère case de Tab dans X
ldab    Index       ; La valeur de l'index dans B
lslb                ; On multiplie l'index par 2 → Tableau de words!
movw    b,x,Data2 ; Data2 = Tab[Index]
```