

LES PSEUDO-INSTRUCTIONS

Les pseudo-instructions sont des commandes à l'assembleur. Toutes les pseudo-opérations commencent par un point. Les pseudo-instructions sont composées soit d'une pseudo-opération seulement, ou alors d'une pseudo-opération suivie par une ou plusieurs expressions (séparés par des virgules). Il existe deux types de pseudo-instructions:

Les pseudo-instructions avec une étiquette éventuelle précédante: .ASCII, .ASCIZ, .ASCIZE, .BLK.n, .DATA.n, .FILL.n, .n (.8, .16, .32, etc.), .STRING, .SYSCALL.n. Ces pseudo-instructions génèrent du code ou réservent de la place mémoire. Pour les mots générés d'une longueur de 16 et de 32 bits l'ordre des octets dépend du processeur.

Les pseudo-instructions sans étiquette précédante: .ALIGN, .APC, .BASE, .CHAP, .ELSE, .END, .ENDIF, .ENDLIST, .ENDMACRO, .ENDTEXT, .ERROR, .EVEN, .EXPORT, .IF, .IMPORT, .INS, .LAYOUT, .LIST, .LISTIF, .LOC, .LOCALMACRO, .MACRO, .MESSAGE, .ODD, .PAGE, .PROC, .PROCSET, .PROCVAl, .RANGE, .REF, .START, .TEXT, .TITLE. Ces pseudo-instructions ne génèrent pas de code et ne dépendent pas du processeur. L'étiquette précédante n'est pas admis à cause des raisons suivantes:

- a) ambiguïté: une étiquette p. ex. devant .LOC laisse un doute sur cette valeur.
- b) étiquette cachée: une étiquette par exemple devant la pseudo-instruction .IF n'apparaît pas dans le listing, car les pseudo-instructions pour l'assemblage conditionnel ne sont pas copiées dans le fichier listing.
- c) impossibilité: une étiquette par exemple devant .MACRO n'a pas de sens.
- d) les pseudo-instructions sont en premier lieu des commandes à l'assembleur et n'ont rien à faire avec les étiquettes, qui sont utilisées dans le programme de l'utilisateur.

.ALIGN

Descript.: Ajuste la valeur de l'APC à la prochaine adresse multiple de la valeur indiquée.

Comment.: Le contenu des locations mémoire sautées n'est pas défini.

Exemple:

```

.LOC    16'234           ; APC a la valeur 16'234
.ALIGN  16'200           ; APC a la valeur 16'400
.ALIGN  16'100           ; APC a la valeur 16'400

```

.APC

Descript.: Choisit un des compteurs d'adresse de l'assembleur (APC).

Comment.: Normalement, on utilise seulement qu'un compteur d'adresse de l'assembleur. Avec la pseudo-instruction .APC le programmeur peut choisir entre plusieurs compteurs d'adresse (p.ex. parties ROM et RAM). Jusqu'à huit valeurs différentes d'APC sont usuelles. Toutes les valeurs d'APC sont mis à zéro au début du programme.

Exemple:

```

ROM    =    0
RAM    =    1
      .APC   ROM           ; ROM: à partir de 16'0
      .LOC   16'0
      .APC   RAM           ; RAM: à partir de 16'2000
      .LOC   16'2000
      ...
      .APC   ROM           ; programme
      JUMP   DEBUT
TESTRAM:
      MOVE.8 ALPHA,D0
      .APC   RAM           ; données
ALPHA: .8    0
      ...

```


.BLK

Descript.: Ajoute à la valeur de l'APC le produit de la taille des données et du nombre.

Comment.: Cette instruction réserve des locations mémoire sans modifier leur contenu. On peut mélanger différentes tailles de données. Ainsi on exprime mieux la structure des données à réserver.

Exemples: .BLK.8 100 ; ajoute à l'APC 100
 .BLK.16 100 ; ajoute à l'APC 200
 .BLK.32 100 ; ajoute à l'APC 400
 .BLK.8.16.32 12 ; ajoute à l'APC 84 (1+2+4)*12

.CHAP

Descript.: Insère le texte indiqué dans le soustitre de l'entête.

Comment.: Cette pseudo-instruction commence une partie importante du programme ou un sous-programme. Le texte indiqué décrit en quelques mots les fonctions réalisées. Le texte apparait dans le champ du soustitre de l'entête. Normalement provoque cette pseudo-instruction un saut de page.

Exemple: .CHAP traitement des symboles

.DATA

Descript.: Insère la valeur donnée dans le programme objet. Chaque valeur a la taille des données mentionnée.

Comment.: Pour les tailles des données de 16 et 32 bits, l'ordre des octets dépend du microprocesseur (défini par .PROC). Le compteur d'adresse de l'assembleur est incrémenté selon la taille indiquée.

Exemples: .DATA.8 255 ; gamme (.8): -256..+255
 .DATA.16 16'FFFF
 .DATA.32 -1 ; = 16'FFFFFFFF
 .DATA.8.8.16 "A", "Z", AD_A_Z

Pseudo-instructions équivalentes: .8, .16, .32, ...

.ELSE

Descript.: Les lignes d'instructions suivantes jusqu'au .ENDIF correspondant sont assemblées, si l'expression du .IF correspondant était fausse (FALSE). Un texte quelconque suivant .ELSE est ignoré par l'assembleur.

Comment.: Le texte qui suit la pseudo-instruction .ELSE est seulement une aide supplémentaire pour le programmeur. C'est surtout le cas, quand les pseudo-instructions .IF, .ELSE et .ENDIF sont très éloignées et imbriquées.

Exemple: voir .IF.

.END

Descript.: Termine les instructions du programme. Des commentaires supplémentaires peuvent se trouver après cette pseudo-instruction.

Comment.: L'assembleur ignore toutes les lignes d'instructions éventuelles après cette pseudo-instruction. La pseudo-instruction .END peut aussi terminer un fichier inséré (avec la pseudo-instruction .INS).

Exemple: .END

.ENDIF

Descript.: Termine une section d'un .IF ou .ELSE. Un texte quelconque suivant .ENDIF est ignoré par l'assembleur.
 Comment.: Le texte qui suit la pseudo-instruction .ENDIF est seulement une aide supplémentaire pour le programmeur. C'est surtout le cas, quand les pseudo-instructions .IF, .ELSE et .ENDIF sont très éloignées et imbriquées.
 Exemple: voir .IF.

.ENDLIST

Descript.: Termine une section de listage conditionnel.
 Comment.: On peut ajouter un texte après la pseudo-instruction .ENDLIST. Ce texte est ignoré par l'assembleur, mais augmente la lisibilité, si l'on répète l'expression de la pseudo-instruction .LIST correspondante. Ceci est surtout conseillé, si le programme est long et/ou les pseudo-instructions .LIST sont imbriquées.
 Exemple: voir .LIST.

.ENDMACRO

Descript.: Termine la définition d'une macro.
 Exemple: voir .MACRO.

.ENDTEXT

Descript.: Retour à des lignes d'instructions.
 Exemple: voir .TEXT.

.ERROR

Descript.: Génère un message d'erreur.
 Exemple: .ERROR Erreur: programme trop long

.EVEN

Descript.: Ajuste la valeur de l'APC à une valeur paire (la valeur n'est pas changée si la valeur est déjà paire).
 Comment.: Le contenu de la cellule mémoire sautée n'est pas défini.
 Exemple: .LOC 16'233 ; APC a la valeur 16'233
 .EVEN ; APC a la valeur 16'234
 .EVEN ; APC a la valeur 16'234

On peut remplacer la pseudo-instruction .EVEN par:
 .ALIGN 2

.EXPORT

Descript.: Définit les symboles exportés.
 Comment.: Normalement on peut seulement exporter des adresses. Le fichier objet généré contient ces informations. L'éditeur de liens (linker) contrôle ensuite, si tous les symboles ont été définis.
 Exemple: .EXPORT MUL64, DIV64

.FILL

Descript.: Remplit la longueur indiquée avec la valeur indiquée.
 Comment.: L'exemple suivant remplit 2048 locations mémoire avec la valeur 16'FF (p.ex. pour une EPROM):
 .FILL.8 2048, 16'FF

Autres exemples:

.FILL.16 10, 16'A5A5
 .FILL.32 16, AD_ROUT

.IF

Descript.: Les lignes d'instructions suivantes jusqu'au `.ELSE` ou `.ENDIF` correspondant sont assemblées, si l'expression est vraie (TRUE). Les `.IF` peuvent être imbriquées.

Comment.: Cette pseudo-instruction permet l'assemblage conditionnel.

Exemple:

```
.IF      M6801
...          ; est assemblé si M6801 = TRUE
  .IF      DEBUG
...          ; est assemblé si M6801 = TRUE
            ; et DEBUG = TRUE
  .ENDIF DEBUG
...
.ELSE      M6801
...          ; est assemblé si M6801 = FALSE
.ENDIF M6801
```

.IMPORT

Descript.: Définit les symboles importés.

Comment.: Les symboles importés doivent être exportés d'un autre module (`.EXPORT`). Les symboles importés sont des adr.

Exemple:

```
.IMPORT MUL64, DIV64
```

.INS

Descript.: Insère le fichier mentionné. Le nom du fichier obtient la même extension comme le fichier source.

Comment.: Pour un programme complexe et de grande taille il est souvent plus commode, de diviser le programme en plusieurs fichiers. Le programme principal insère ensuite ces sousprogrammes séparés. De plus, on pourrait réunir des constantes souvent utilisées dans un fichier.

Exemple:

```
.INS      CONSTANTES
```

.LAYOUT

Descript.: Définit les paramètres pour l'aspect général du listage. Ces paramètres sont indiqués ou bien par un fichier ("`FILE fichier`") ou alors par une liste de noms réservés (`OCT`, `HEX`, `TAB n`, `LENGTH n`, `WIDTH n`, etc.).

Comment.: Cette pseudo-instruction a été définie d'une manière assez générale et permet toutes les extensions possibles. Avec qu'une pseudo-instruction on veut définir aussi flexiblement que possible l'aspect du listage. Les autres assembleurs définissent souvent un bon nombre de pseudo-instructions différentes, qui sont réunies ici dans une seule pseudo-instruction.

Les noms suivants ont été réservés jusqu'à présent:

```
HEX, OCT  (données sous forme sexadécimale ou octale)
LENGTH n  (définit n lignes par page; n=0: infini)
WIDTH n   (définit n caractères par ligne)
TAB n     (un tabulateur correspond à n espaces)
LEADING0 TRUE/FALSE (avant adr./val. [.LST]: '0'/' ')
DEFFIRST, LSBFIRST, MSBFIRST (ordre [.16, .32, ...])
```

Exemple:

```
.LAYOUT HEX, LENGTH 66
```

.LIST

Descript.: Les lignes d'instructions suivantes sont copiées dans le listage, si l'expression est vraie (TRUE). Les pseudo-instructions `.LIST` peuvent être imbriquées.

Comment.: Cette pseudo-instruction permet le listage conditionnel.

Exemple:

```
.LIST      DEBUG
            ; cette partie du programme apparait si DEBUG = TRUE.
. ENDLIST DEBUG
```

.LISTIF

Descript.: Affiche toutes les pseudo-instructions **.IF/.ELSE/.ENDIF** dans le listage (**.LST**).

Comment.: Si l'il n'y a pas d'expression ou si l'expression est non nulle, alors l'affichage est fait.

Exemple: `.LISTIF DEBUG_IF`

.LOC

Descript.: Assigne une nouvelle valeur au compteur d'adresse de l'assembleur courant (APC). Les valeurs de l'APC sont mis à zéro au début du programme.

Exemple: `.LOC 16'2000`

.LOCALMACRO

Descript.: Déclare les symboles locaux dans une macro.

Comment.: Cette pseudo-instruction est seulement valable dans une définition d'une macro. L'assembleur remplace les symboles correspondants par des symboles locaux (p.ex. **M_10\$**) au moment de l'appel à cette macro.

Exemple: voir **.MACRO**.

.MACRO

Descript.: Commence une définition d'une macro avec le nom de la macro et une liste facultative de paramètres. La syntaxe pour le nom de la macro est la même que pour les symb.

Comment.: La liste de paramètres définit les paramètres par défaut. Ils sont utilisés, s'il n'y pas de paramètres au moment de l'appel à cette macro. Un paramètre est utilisé à l'intérieur d'une macro par le signe pourcent, suivi d'un chiffre.

Exemple (68000): Définition de la macro:

```
.MACRO DELAI, D0, 1
  .LOCALMACRO BOUCLE
  MOVE.16 #%2-1,%1
BOUCLE: DJ.16,NMO %1,BOUCLE
  .ENDMACRO
```

Premier appel:

```
DELAI D1,UNE_MSEC
```

produit:

```
MOVE.16 #UNE_MSEC-1,D1
```

```
M_0$: DJ.16,NMO D1,M_0$
```

Deuxième appel:

```
DELAI ,DEUX_MSEC ; sans premier paramètre
```

produit:

```
MOVE.16 #DEUX_MSEC-1,D0
```

```
M_1$: DJ.16,NMO D0,M_1$
```

.MESSAGE

Descript.: Affiche le texte pendant l'assemblage.

Exemple: `.MESSAGE assemblage dure environ 10 minutes`

.n (.8, .16, .32, ...)

Descript.: Insère la valeur donnée dans le programme objet. Chaque valeur a la taille des données mentionnée.

Comment.: Pour les tailles des données de 16 et 32 bits, l'ordre des octets dépend du microprocesseur (qui a été défini par **.PROC**). Le compteur d'adresse de l'assembleur est incrémenté selon la taille.

Exemples: `.8 255 ; gamme (.8): -256..+255`
`.16 16'FFFF`
`.32 -1 ; = 16'FFFFFFFF`
`.8.8.16 "A", "Z", AD_A_Z`

