# *Tp n°1*

# *La syntaxe d'un programme TASM*

# *Structure d'un programme Tasm*

**<nom-pile>** *segment stack*
<taille de la pile>
**<nom-pile>** *ends*

**<nom-seg-données>** *segment*
<déclarations des variables, des constantes, tableaux>
**<nom-seg-données>** *ends*

*Assume ds* : data, *cs* : code, *ss* : pile
*; ceci est un commentaire*

*; le code commence ici*
**<nom-seg-code>** *segment*
**<étiquette>**:
les instructions…..

**<nom-seg-code>** *ends*
*end* **<étiquette>**

# *Exemple*

```
pile segment stack
      db 010h dup (?)
pile ends
data segment
       texte db "ceci est une chaine de caracteres",'$'
data ends
assume ds : data, cs : code, ss : pile

;  le code commence ici
code segment
debut:   mov ax, data          ; fixer le registre segment de donnee
         mov ds, ax            ; charger ds par l'adresse de la chaine
         mov dx, offset texte  ; reperer le début
         mov ah, 09h           ; preparer l'interruption d'affichage du texte
         int 21h               ; appeler l'interruption
         mov ah, 4CH           ;  Retour au DOS
         int 21H
code ends
end debut
```

# *La syntaxe d'une opération*

- Une opération en tasm peut avoir la structure suivante:

Généralement:

**a.**

| opération | 1ier opérande | 2ème opérande |
|---|---|---|

Exemple: ***mov ax, 'a'***

**b.** Ou :

| opération | L' opérande |
|---|---|

Exemple : ***int 21h***

**c.**Ou :

| opération |
|---|

Exemple: ***ret***

# *La syntaxe d'une opération*

- Les opérations sont prédéfinies par le langage

Exemples: *add mul div sub dec jmp…*

- Les opérandes peuvent être des valeurs immédiates ou contenues dans des registres ou variables

Exemples

```
add  ax, 2
add  ax, bx
add  ax, moy
```

***Remarque***: jamais deux variables ou deux valeurs

# *Déclaration d'une variable*

- Pour chaque variable, il faut spécifier**:**
  - son nom
  - sa taille (pseudo-instruction)
  - son contenu initial éventuel

- *Exemple :*

```
moy  db  12
moy db  ?    (* valeur indéfinie)
```

# *Déclaration d'une variable*

- **Le nom**
  - ne peut contenir que des lettres ou des chiffres (et "_")
  - doit commencer par une lettre
  - doit avoir une taille de 8 caractères maximum
- **La taille**

  **db : declare byte** (1 byte)
  **dw : declare word** (2 bytes)
  **dd : declare double** (4 bytes)
- **La valeur initiale**

  on est obligé d'initialiser une valeur. Si la valeur n'est pas connue dans ce cas on met un "?".

# *Déclaration d'une variable*

## Déclaration des variables de type caractères:

**DB ou db** est aussi utilisée pour définir des chaînes de caractères quelque soit leur taille

- Une chaîne de caractères doit se trouver entre apostrophes
- si elle est destinée à l'affichage (sur l'écran), elle devra être terminée par le signe '$'

Exemple

*texte* **db "**ceci est une chaîne de caractères**",*'$'***

# *Déclaration d'un tableau*

La syntaxe:

<nom-tableau> <type> <taille> [*dup*] (valeurs)

* *dup* : signifie dupliquer ou répéter l'opération qui le précède

**Exemples** :

- tab1 db 10 dup (0)
- tab2 dd 5 dup (0, 1, 2, 3, 4)

  ou tab2 dd 5 0, 1, 2, 3, 4
- tab3 dw 100 dup (?)

# *À faire*

- Copier le code de l'exemple cité au début
- Compiler
- Lier
- Exécuter
- Appeler le turbo-debugger (td) utiliser le mode trace (F7)
  - observer l'état de la mémoire
  - les valeurs des registres
  - les flags