

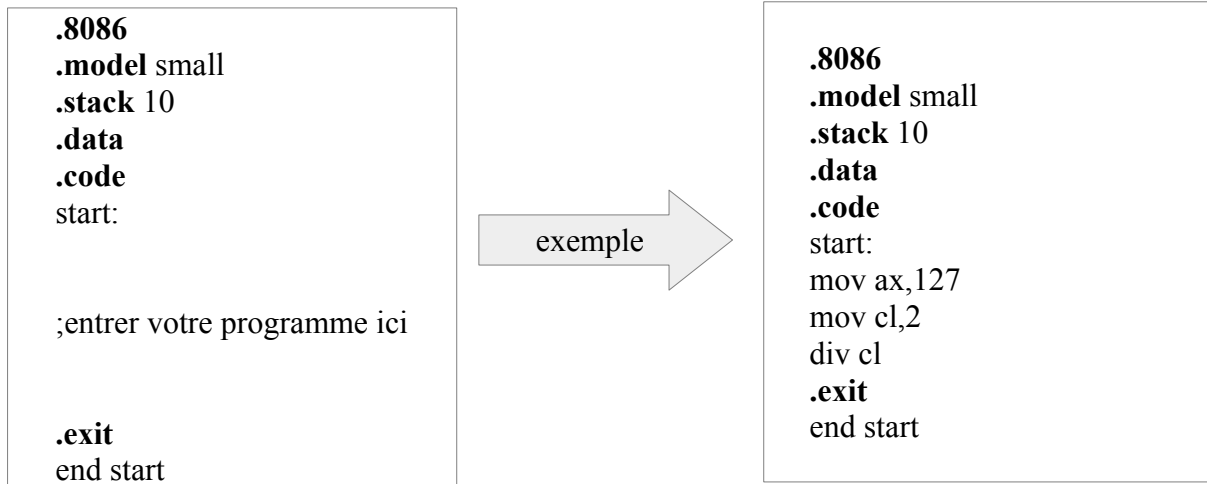
UNIVERSITE AMAR TELIDJI DE LAGHOUAT

Département d'Informatique
1ere Master Informatique

Serie n°4 Tasm (Turbo Assembleur)

1) Structure d'un programme en Tasm

La structure d'un programme en Tasm est la suivante :



-Ecrire en tasm le programme qui permet de faire le decalage à gauche.

2) Declaration des données

*Les variables sont déclarées apres la directive `.data`. Avec les outils suivants :

`EQU/DB/DW/DD/DF/DP/DQ/DT`

`VAL EQU 50 ; assigne la valeur 50 au nom VAL`

`ET1 EQU VAL* 5 + 1 ; assigne une expression calculer a VAL`

`Vil DB 12H ; Définit une variable (un octet) de valeur Initiale 12.`

`Tab DB 18H, 15H, 13H ; définit un tableau de 3 cases
; (3 octet) Qui démarre à partir de l'adresse TAB.`

`Mess DB 'ISET' ; définit aussi un tableau mais les valeurs de chaque case
; n'est autre que le code ascii de chaque lettre.`

`Name DB ? ; définit une variable 8 bits de valeur initiale quelconque .`

VIL	12
TAB	18
	15
	13
Mess	'I'
	'S'
	'E'
	'T'
Name	05

← Valeur quelconque

Directive	Meaning
DB	Define byte-size data.
DW	Define word-size data.
DD	Define doubleword-size data.
DQ	Define quadword-size data.
DF	Define 48-bit 80386 far-pointer-size (6 byte) data.
DP	Define 48-bit 80386 far-pointer-size (6 byte) data.
DT	Define tenbyte (10-byte) size data.

*Directive dup Lorsque l'on veut déclarer un tableau de n cases, toutes initialisées à la même valeur, on utilise la directive *dup*:

```
tab DB 100 dup (15) ; 100 octets valant 15
```

```
y DW 10 dup (?) ; 10 mots de 16 bits non initialises
```

3) L'adressage

*Bien qu'il soit possible de n'utiliser qu'un seul segment à tout faire, la plupart des programmes EXE ont un segment réservé au code, un ou deux autres aux données, et un dernier à la pile. Les segments sont indiqués dans les registres : CS,DS,SS,ES.

*En général le déplacement est ajouté par défaut avec le registre segment DS mais il faut signaler qu'on peut utiliser ce mode d'adressage avec d'autres registres segment tel que ES par exemple

```
MOV AX,[adr] ;ds par default
```

```
MOV AX, ES : [adr] ;es
```

*L'adresse de l'opérande peut être stockée dans un registre de base (BX ou BP) ou d'indexe (SI ou DI).

```
MOV BX,offset adr
```

```
MOV AX,[BX]
```

```
MOV AX,[BX]+2
```

```
MOV AX,[BX+2]
```

```
MOV AX,2[BX]
```

```
mov ax,ds:[di]
```

*Les directives Word PTR et Byte PTR :

Dans certains cas, l'adressage indirect est ambigu. Par exemple, si l'on écrit :

```
MOV [BX], 0 ; range 0 à l'adresse spécifiée par BX
```

L'assembleur ne sait pas si l'instruction concerne 1, 2 ou 4 octets consécutifs. Afin de lever l'ambiguïté, on doit utiliser une directive spécifiant la taille de la donnée à transférer :

```
MOV byte ptr [BX], val ; concerne 1 octet
```

```
MOV word ptr [BX], val ; concerne 1 mot de 2 octets
```

-Écrire le programme qui permet de copier le mot bonjour à la place du mot bonsoir et vis versa

EXERCICE

Instruction	Codage	
MOV AL, [<i>adr</i>]	A0 <i>adr</i>	Lis l'emplacement mémoire <i>adr</i> et le charge dans AL.
MOV [<i>adr</i>], AL	A2 <i>adr</i>	Stocke la valeur de AL dans la mémoire d'adresse <i>adr</i> .
ADD AL, [<i>adr</i>]	02 06 <i>adr</i>	Ajoute dans AL la valeur lue à <i>adr</i> .

- 1- Combien d'octets occupent chacune de ces instructions ?
- 2- Écrire avec ces instructions un programme qui ajoute le contenu des cases mémoires d'adresses 130H et 131H, puis range le résultat à l'adresse 132H.
- 3- Donner l'adresse de chaque instruction du programme, sachant que la première instruction est implantée à l'adresse 1000H.
- 4- on supposera que la valeur rangée à l'adresse 0130H est 88H, et la valeur 5 en 0131H.
- 5- Quel est le résultat de l'addition si l'on a les valeurs 254 et 10 rangées en 0130H et 0131H ?