Pr: Ismaili Alaoui El Mehdi Filière: SMI-4

Architecture des ordinateurs

Chapitre 7: Macros & Sous-programmes

1 Les macros

Il est possible de choisir pour certaines suites d'instructions qui se répètent constamment dans un programme un nom qui les représente. Ces instructions forment une **macro**.
L'écriture de macro est un moyen pratique de rendre le code source plus rapide à écrire et plus lisible.
Lorsque l'assembleur rencontre le nom d'une macro dans le code source, il le remplace par le code de la macro. *Tout se passe exactement comme si vous aviez tapé vous-même ce code à la place du nom de la macro*.

1. 1 Déclaration et appel d'une macro

Nom_Macro **macro** paramètres; Indique le début d'une macro appelée *Nom_Macro*.
instruction 1
instruction 2
...
End**m** ;Indique la fin de la définition d'une macro.
Dans le segment du code, on appelle cette macro comme suit : Nom_Macro args
À l'aide de cette macro, le programme met d'abord args dans paramètres.

❒ Remarque: En général, les macros sont mises après le segment de pile.

❒ Exemple : Afficher une chaîne de caractères.
Afficher macro chaîne
mov dx, offset chaîne
mov ah, 09h
int 21h
endm
Dans le segment du code, Si on appelle cette macro comme suit : Affiche msg
Le programme met msg dans chaîne.

2 Les sous-programmes

- ❒ Les sous-programmes jouent, en assembleur, le rôle des procédures et des fonctions dans les langages de haut niveau (C++, java,...).
- ❒ Elles structurent les programmes en donnant un nom à un traitement et en réduisant la taille des programmes qui utilisent plusieurs fois la même séquence de code.
- ❒ L'utilité des sous-programmes est la même que dans les autres langages, c-à-d «écrire une fois, utiliser plusieurs fois».
- ❒ Les sous-programmes permettent une programmation plus souple et plus organisée en langage assembleur.

2. 1 Déclaration d'un sous-programme

Etant donnée qu'un sous-programme est un suite d'instructions, il s'agit de regrouper les instructions composant le sous-programme entre des mots clés. L'ensemble de cette manipulation est appelée déclaration de sous-programme.

Nom **proc near**
    instruction 1
    instruction 2
    ...
    Ret
End**p** ; Indique la fin de la procédure.
near signale que le sous-programme est situé dans le même segment que le programme appelant.

- ❒ Remarque 1: En général, les sous-programmes sont mis à la fin du programme principal.
  Exemple :

```
...
Fin: mov ah,4ch
     int 21h
Afficher proc near
        mov ah, 09h
        mov dx, offset message
        int 21h
        ret
Endp
Code ends
end Main
```

❒ Remarque 2:
On peut aussi les mettre dans la partie du segment du code. Seulement, il faudra s'assurer que la première instruction du code exécutée soit celle du programme principal. Pour cela, il suffit juste de mettre un **jmp** juste avant la déclaration du sous-programme.
Exemple :

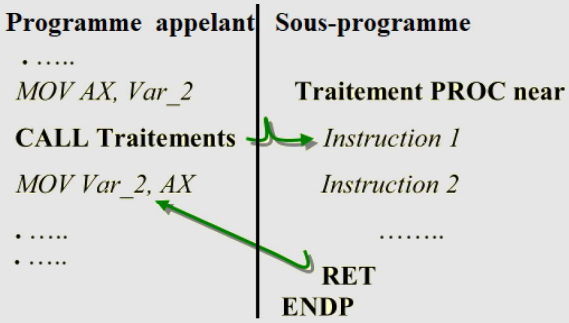
```
...
Code segment
    jmp Main
    Afficher proc near
        mov ah, 09h
        mov dx, offset message
        int 21h
        ret
    Endp
Main: Assume cs:code, ds:data, ss:pile
    mov ax,data
    ...
Fin: mov ah,4ch
    int 21h
Code ends
end Main
```

2. 2 Appel d'un sous-programme

Nous décrivons ici l'instruction d'appel d'un sous-programme: call Nom_de_sous-programme.

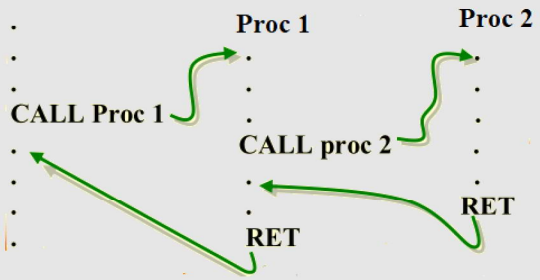
❒ Remarques :

❒ Les sous-programmes doivent être terminés (juste avant l'instruction Nom Endp) par l'instruction RET.



Après RET, le processeur exécute l'instruction MOV Var_2, AX.
Donc, l'instruction RET est considérée comme une instruction de retour vers le programme appelant.

❒ Les appels imbriqués sont également possible en assembleur en utilisant le même principe d'appel et de retour.



❒ call est une nouvelle instruction de branchement inconditionnel.

❒ En assembleur, Nom du sous-programme désigne une adresse de début du sous-programme sur 2 octets.

2. 3 Passage de paramètres

2. 3. 1 Passage de paramètres par registre

Les paramètres à utiliser par le sous-programme sont envoyés via les registres. Autrement dit, ces paramètres sont stockés dans les registre avant l'appel sous-programme.

❒ Exemple 1: La permutation du contenu de deux registres.

Programme appelant Sous-programme

.....

.....

.....

.....

.....

.....

❒ Exemple 2: Calculer la factorielle d'un nombre dont la valeur est passée en entrée dans le registre AX et ranger le résultat dans BX.

Programme appelant Sous-programme

.....

2. 3. 2 Passage par référence

Les tableaux et les chaînes de caractères peuvent être passés comme paramètres lors des appels de sous-programmes, ce passage s'effectue par adresse (par référence).

❒ Exemple: Afficher le contenu d'un tableau.

Programme appelant | Sous-programme

2. 3. 3 Passage de paramètres dans la pile

Cette méthode de passage de paramètres consiste à stocker les valeurs des paramètres dans la pile avant l'appel de sous-programme (grâce à l'instruction PUSH), puis de lire le contenu de la pile grâce à un registre spécial (BP: Base pointer) qui permet de lire des valeurs dans la pile sans les dépiler, ni modifier le pointeur de sommet de pile (SP).
L'appel de sous-programme se fera comme suit:

- ❒ PUSH parametre1
- ❒ PUSH parametre2
- ❒ ....
- ❒ CALL sous-programme

Le sous-programme commencera par l'instruction suivante:

- ❒ MOV BP, SP ; BP pointe sur le sommet de la pile

- Puis pourra contenir des instructions du type:
  - MOV AX, [BP] ;Stocke le contenue de sommet de la pile dans AX, sans dépiler
  - MOV BX, [BP+2] ;Stocke le mot suivant de la pile dans BX, sans dépiler
  - ...

- Exemple 1: La permutation du contenu de deux registres.

Programme appelant	Sous-programme

- Exemple 2: Calculer le PGCD de deux nombres passés en paramètre..

Programme appelant	Sous-programme

On suppose que l'on dispose du sous-programme affiche_nombre qui affiche le résultat un nombre à l'écran.

Le passage de paramètres par pile offre la possibilité d'utiliser un nombre plus grand de paramètres. Cette méthode permet également de sauvegarder les résultats de sous-programme dans la pile ou également dans les registres processeur.

❒ Remarques:
— Les macros, à la différence des sous-programmes, n'ont aucune signification pour la machine. Seul le compilateur comprend leur signification. Elles ne sont qu'un artifice mis à la disposition du programmeur pour clarifier son programme.
— Ainsi, si vous appelez quinze fois une macro dans votre programme, le compilateur écrira quinze fois le code de cette macro. C'est toute la différence avec les sous-programmes qui ne sont écrites qu'une seule fois mais peuvent être appelés aussi souvent qu'on veut à l'aide d'un call.