Université de Tunis El-Manar Faculté des Sciences de Tunis Département des Sciences de l’Informatique

IF3

**Mini Projet Assembleur N2**

*Projet développer par : Aymen Ben Mhamed(Amphi B GR03) Ayda Rekaya (Amphi A GR03)*

***Section : Ingéniorat en Informatique IF3   
Matière : Système d’Exploitation***

***TP système d’exploitation N°2 :***

***Mini Projet Assembleur***

***Détournement d’une interruption***

***Sommaire :***

|  |  |
| --- | --- |
| **Page 1** | ***Sommaire*** |
| **Page 3** | * ***Introduction à l’assembleur*** * ***Introduction à l’utilisation du Macros*** * ***Introduction aux Fichiers en assembleur*** |
| **Page 8** | ***Problématique*** |
| **Page 9** | ***Programme assembleur***  ***Code source (TP2\_Détournement d’une interruption « int 20h »)*** |
| **Page 19** | ***Étape de passage du code source en code exécutable*** |
| **Page 21** | ***Conclusion*** |

***Section : Ingéniorat en Informatique IF3   
Matière : Système d’Exploitation***

***TP système d’exploitation N°2 :***

***Mini Projet Assembleur***

***Détournement d’une interruption***

***Introduction à l’assembleur :***

*Contrairement aux langages évolués, l'assembleur, ou « langage d’assemblage » est constitué d'instructions directement compréhensibles par le microprocesseur : c'est ce qu'on appelle un langage de bas niveau. Il est donc intimement lié au fonctionnement de la machine. C'est pourquoi il est relativement difficile à assimiler, en tout cas beaucoup plus que les langages de haut niveau.*

*Cela explique également pourquoi il existe au moins autant de langages d’assemblage que de modèles de microprocesseurs*

***Introduction à l’utilisation des Macros :***

*Un programme sera souvent constitué de plusieurs sous-programmes. Cela facilite la lisibilité et la clarté du code.*

*Un macro (ou sous routine ou en encore sous programme) permet de cataloguer un algorithme (Procédure ou fonction ou macro) pour pouvoir ensuite faire appel à cet algorithme à tout moment (appel de procédure ou de fonction ou appel de macro).*

*Les sous-programmes sont en général prévus pour exécuter un traitement particulier.*

*En règle générale, les sous-programmes constituent une excellente méthode de découpage de gros programme en petites parties beaucoup plus faciles à manipuler, il ne faut donc pas hésiter à les utiliser.*

**LES INTERRUPTION :**

*Il existe de type des interruptions :*

*INTERRUPTIONS LOGICIELLES & INTERRUPTIONS MATERIELLES.*

*Dans ce TP on s’intéresse aux celles LOGICIELLES.*

***Interruption logicielle :***

Les   interruptions   logicielles   sont   semblables   aux   interruptions matérielles.  L'unique  différence  réside  dans  le  fait  que  les  interruptions logicielles sont émises par des programmes. Les cinq premières interruptions sont définies par Intel. Les autres interruptions sont définies par le DOS et le BIOS. Ces interruptions ont une fonction définie, par exemple la

lecture et l'écriture sur le disque, l'écriture des données à l'écran, etc. Contrairement  à  l'entrée  INTR  du  microprocesseur,  l'interruption logicielle ne peut être ni invalidé ni masquée.

**Exemple d'interruption software :**

*Fonctions BIOS :*

Int 1Ah, Fct 02h  Date et heure : Lecture de l'horloge temps réel BIOS (> AT) : Cette fonction permet de lire l'heure de l'horloge temps réel alimentée par batterie. Comme ce type d'horloge n'existe que sur les AT, seul ce modèle de PC soutient cette fonction

Fonctions Gestionnaire de souris

Int 33h, Fct 00h    Réinitialisation du driver de souris Initialise le driver de souris et enclenche ainsi le test de la souris

**Problématique :**

*On se propose d’effectuer un programme assembleur qui permet d’effectuer les différentes tâches suivantes :*

1. *Modifier la description originale de l’interruption int20h par une nouvelle routine spécifiée.*
2. *Développer Une New\_Routine qui prise en charge la nouvelle gestion de l’interruption int20h .*
3. *Informer l’utilisateur du succès du détournement de l’interruption int20h , en lui affichant un message.*
4. *Expliciter la Routine de détournement de l’interruption int20h.*
5. *Afficher un message de « Fin de programme » afin d’expérimenter la Routine de Détournement de l’interruption int20h.*

* *Effectuer les 5 tâches proposées chacune dans un sous programme (macro) indépendant puis appeler toutes les macros dans le programme principal.*

Code source

title mminiprojet2

.model small

.stack 100h

.386

.data

msg1 db 0dh,0ah,"SUCCES DE DEROULEMENT DE L INTERRUPTION int 20h ",'$'

msg3 db 0dh,0ah,"verifier votre choix svp!!!!",'$'

msg4 db 0dh,0ah," programme va se fermer ",'$'

break db 10,13,"appuyer sur une touche pour sortir","$"

message2 DB 0dh,0ah," appuyer sur (y) pour continuer (n) pour quitter : ","$"

.code

**fin\_propre macro**

;restaurer l'ancien vecteur d'interruption

pop es

pop bx

mov al,20h

mov ah,25h

int 21h

mov dx,bx

int 21h

mov bx,es

int 21h

;appel systeme de l'interruption 20h

int 20h

**fin\_propre endm**

;=====================================

**routine\_detournement proc**

debut:

MOV AH,9

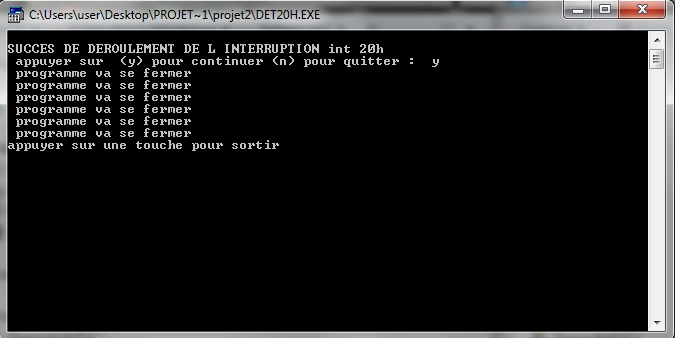
LEA DX,message2

int 21h

Mov AH,1

Int 21h

cmp al,'n'

**

jne continue

fin\_propre

iret

continue:

cmp al,'y'

jne debut

call aff\_msg

ret

**routine\_detournement endp**

;============================================

**aff\_msg proc**

;enregistrer ds cx le nbr de fois a repeter le msg

mov cx,5

boucle:

mov ah,09

mov dx,offset msg4

int 21h

loop boucle

ret

**aff\_msg endp**

;=======================================

**success\_det macro msg**

mov ah,09

mov dx,offset msg1

int 21h

endm

new\_routine proc

success\_det msg

call routine\_detournement

ret

**new\_routine endp**

**;=====================================**

**redirect\_int macro new\_routine**

;lecture du vecteur d'interruption

mov al,20h

mov ah,35h

;sauvgarder les vecteur reels de l'int 20h

push bx

push es

lea dx,new\_routine

mov al,20h

mov ah,25h

int 21h

**redirect\_int endm**

;===============================

main proc

mov ax,@data

mov ds,ax

redirect\_int new\_routine

call new\_routine

int 21h

mov ah,09h

mov dx,offset break

int 21h

mov ah,1

int 21h

mov ax,4c00h

int 21h

.exit

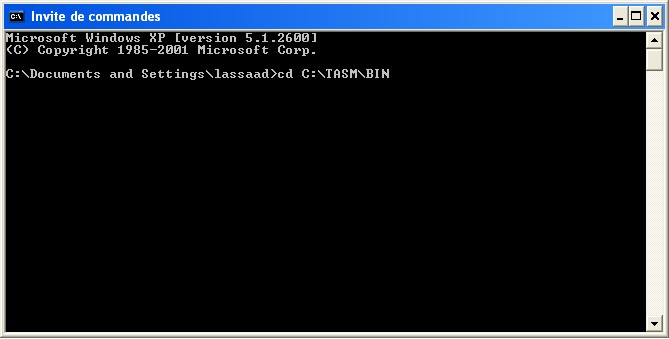
main endp

end main

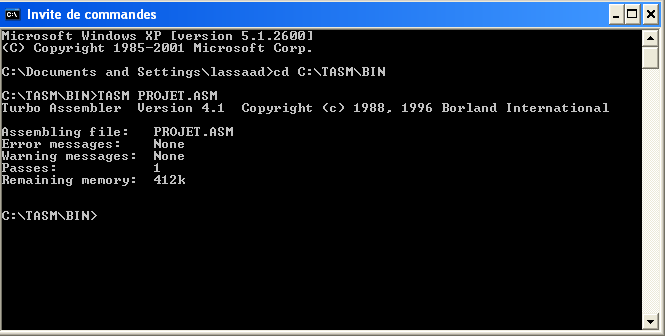
Étape de passage du code source en code exécutable

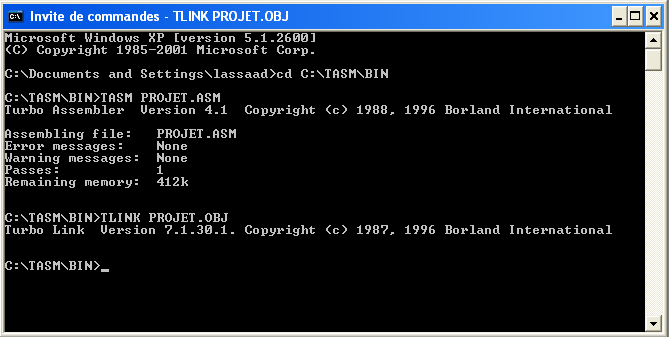
*Pour créer un programme assembleur il faut :*

* *Créer le programme au langage assembleur avec l’extension ″ .ASM ″.*
* *Ouvrir l’invite de commande (MS-DOS) pour effectuer les taches de compilations et de gestion des liens.*
* *En compilant avec l’utilisation de l’instruction ″tasm projet.asm″, le logiciel de compilation vérifie les erreurs et il les localise dans le programme, si le compilateur ne trouve aucune erreur dans le programme alors il crée un fichier portant le même nom avec l’extension ″ .obj ″ qui porte un programme presque converti en code exécutable ( ne converti pas les instruction d’appel des fonctions externes au programme source principal).*
* *La gestion des liens se fait par la balise ″tlink projet.obj″ qui assure la liaison entre les liens des fonctions utilisées dans le programme source et les fonctions définies ailleurs d’où il crée un fichier d’extension ″.map″ pour assurer la liaison les liens puis il crée le fichier exécutable.*



Changement de répertoire vers ″ C:\TASM\BIN ″



Compilation par ″TASM projet.asm ″ et création de fichier ″ PROJET.OBJ 

Gestion des liens par ″ TLINK projet.obj ″ et création de fichier ″ projet.map ″ et d’exécutable ″ projet.exe ″

CONCLUSION

*L’analyse modulaire facilite, d’un part, les tâches de création d’un programme et le décompose, d’autre part, en des sous programmes plus simples à résoudre et montre, en plus, la synchronisation des idées du programmeur.*

*Mais, le problème dans le traitement de l’analyse modulaire est le manque de la rapidité dans surtout le cas des traitements basés sur les boucles (Tel est le cas des chaînes de caractères et des tableaux) parce qu’il traîne au moins une boucle dans un chaque module.*