

RAPPORT DE TP3

Interruptions et programmation de quantum de temps à l'aide de l'interruption périodique 1CH

Réalisée par:

-Nour Saidani

Email:saidaninour@gmail.com

Section :ACAD A

Groupe :03

-Mouna Boubakir

Email:boubakirmw@gmail.com

Section :ACAD A

Groupe :03

Partie I:

```
data segment
    newline db 10,13,'$'
    prompt1 db "Entrez le numero d'interruption: $"
    msg db 'Le resultat est :$'
    buffer db 5 dup (' ') ; Augmentez la taille du buffer pour les
    saisies plus longues
    chaff db 5 dup(' '), '$'
    pkey db "0123456789ABCDEF" ;
    nombre dw ?
    decalage db ?
data ends
```

```
File segment stack
    dw 256 dup(?)
    tos label word
File ends
```

```
code segment
assume cs:code, ds:data,ss:Pile ;cette ligne est essentielle
```

```
StringNumber proc near
    mov ax, data
    mov ds , ax
    mov dx,offset buffer
    mov ah,0Ah
    int 21h
    ret
```

```
StringNumber endp
```

```
String2Number proc near
```

```
    q:
        mov dx, 10
        mul dx
        mov dl, buffer[di] ; Charger le caract?re actuel de la cha?ne de
        caract?res
        sub dl, 30h ; Convertir le caract?re ASCII en nombre
        add ax, dx
        inc di ; Avancer au caract?re suivant
        loop q
    ret
```

```
String2Number endp
```

```
afficherEnHexa proc near
```

```
    push bx
    push cx
    push ax
```

```

    push si
    mov cx, 0000h
    lea si, chaff                ; Adresse du tableau de caract?res
hexad?cimax
    mov decalage, 12
    mov dx, nombre              ; Charger l'adresse du nombre ? afficher
dans DX
rep:
    mov ax, dx                  ; Charger le nombre de 16 bits dans AX
    mov cl, decalage            ; Nombre de d?calages ? effectuer
    shr ax, cl                  ; D?caler ? droite AX par CL bits
    and ax, 0Fh                 ; Conserver uniquement les 4 bits de
poids faible
    lea bx, pkey                ; Adresse du tableau de correspondance
hexad?cimal
    xlat                        ; Charger le caract?re hexad?cimal
correspondant dans AL
    mov [si], al                ; Stocker le caract?re dans le tableau
chaff
    inc si                      ; Avancer au prochain emplacement dans le
tableau
    sub decalage, 4              ; D?cr?menter le compteur de
d?calages
    cmp cl, 0                   ; V?rifier si on a termin? tous les
d?calages
    jnz rep
    mov dx, offset chaff
    mov ah, 09h
    int 21h
    pop si
    pop ax
    pop cx
    pop bx                      ; Si non, r?p?ter le processus
    ret
afficherEnHexa endp

afficherVingsVecteurs proc near
    mov cx, 20
    mov ax, data
    mov ds, ax
    mov bp, sp
    mov bx, 0
    mov ax, 0
    mov al, [bp+2]              ; Charger l'index du vecteur dans AL
    ; Multiplier l'index par 4 pour obtenir l'offset
    mov bl, 4

```

```

    mul bl
    mov bx , ax
repeter:
    push ds ;Empiler le ds pou ne pas le perdre
    ; Rendre le ds 0 pour accder a la tab;e des vecteur
    mov ax , 0
    mov ds , ax
    xor ah , ah ; Pour initiliser le ah a 0
    ; Utilisation de xlat pour charger dans si le ip de la routine
d'int ? partir de la table des vecteurs
    xlat ; Charger le premier octet
    mov dl , al ; Comme dans nos mchine les donnees sont
sauvgarder selon le littel indian alors on charge l'octet faible
    inc ah ; Passer au octet suivant
    mov al , ah
    xlat ; Charger le deuxieme octet
    mov dh , al
    inc ah
    mov si , dx ; Sauvgarder le resultat dans si
    ; Utilisation de xlat pour charger dans dx le cs de la routine
d'int ? partir de la table des vecteurs (le mm traitement)
    xlat
    mov dl , al
    inc ah
    mov al , ah
    xlat
    mov dh , al ; En suite l'octet fort

    pop ds ; Revenir au data segment
    mov nombre , dx
    call near ptr afficherEnHexa
    mov ah , 02h
    mov dl , ':'
    int 21h
    mov nombre , si
    call near ptr afficherEnHexa
    mov dx , offset newline
    mov ah , 09h
    int 21h
    add bx , 4 ; Passer au vecteur suivant
    loop repeter

ret
afficherVingsVecteurs endp
start:
    ; Initialiser les registres de segment
    mov ax , data

```

```

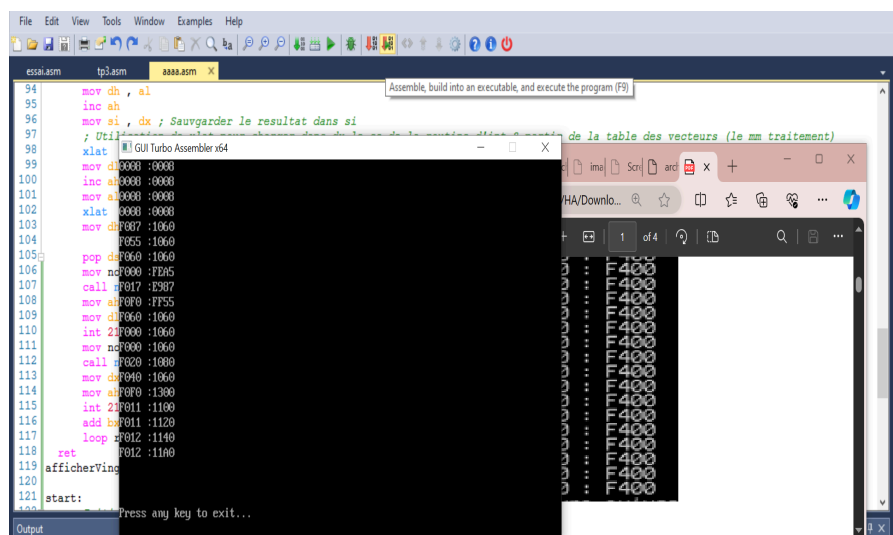
mov ds, ax
mov di, 0
; Initialiser le compteur de boucle ? 0
mov cx, 00h
; Initialiser le registre de pile (SS) et le pointeur de pile (SP)
mov ax, Pile
mov ss, ax
mov sp, offset tos
; Afficher le message demandant pour donner le num d'int
mov dx, offset prompt1
mov ah, 09h
int 21h

; Lire le premier nombre ? partir du clavier
call near ptr StringNumber
mov bx, 0000h
mov bl, buffer[1] ; Charger la taille du nombre dans BL
mov buffer[bx+2], '$' ; Terminer la cha?ne avec un caract?re de
fin de cha?ne
mov cl, buffer[1] ; Charger la taille du nombre dans CL
mov di, 0002h ; Initialiser DI ? 2 pour pointer vers le
premier caract?re du nombre
mov ax, 0000h
call near ptr String2Number
mov bx, 0000h
; Passer le param?tre de la proc?dure (NUM D'INT)
push ax
mov dx, offset newline
mov ah, 09h
int 21h
; Appeler la proc?dure pour afficher les vecteurs d'interruption
call near ptr afficherVingsVecteurs

; Terminer le programme
mov ax, 4c00h
int 21h
code ends
end start

```

Exécution: Ce déroulement est pour N=1.



Partie II:

1- Etudier l'interruption 1CH :

L'interruption 1CH (également connue sous le nom d'INT 1CH ou interruption du clavier à 10 touches) est une interruption logicielle introduite dans les premiers ordinateurs personnels. Elle était principalement utilisée par DOS et les systèmes d'exploitation plus anciens et souvent utilisée pour implémenter des minuteries logicielles ou des tâches planifiées. Elle est appelée par la routine d'interruption numéro 8H qui est déclenché par le trimmer. Cette routine n'a aucune utilité cela pour pouvoir la dérouté et l'utiliser pour le besoin de l'utilisateur.

2- Implémentation de l'interruption 1CH :

Pour implémenter l'interruption 1CH, on doit écrire une routine d'interruption et la charger dans le vecteur d'interruption approprié. Voici les étapes générales pour implémenter l'interruption 1CH :

- Définir la routine d'interruption qui sera exécutée lorsque l'interruption 1CH se produit.
- Charger l'adresse de la routine de service d'interruption dans le vecteur d'interruption 1CH en utilisant l'interruption 25H/21H (Installation d'un vecteur).
- Activer les interruptions pour permettre au BIOS de déclencher l'interruption 1CH.

Une fois ces étapes réalisées, à chaque appelle à la routine d'interruption 8H (chaque 55ms) ce nouveau traitement sera exécuter.

3- Programme 1 :

```
data SEGMENT
    msg1 db "***Debut du quantum de temps logiciel***", 0ah, 0dh,
    "$"
    msg2 db "1 sec ecoulee ...$"
    msg3 db "Derouteent fait...", 0ah, 0dh, "$"
    newline db 13, 10, '$'
    compt db 18 ; Compteur 1000/55 = 18.1
data ENDS
ma_pile SEGMENT STACK
    dw 256 dup(?)
    tos label word
ma_pile ENDS
code SEGMENT
    assume CS:code, DS: data
; Définition de la procédure de déroutement pour l'interruption 1CH
derout_1CH PROC NEAR
    derout:
        mov ax, seg new ; Charger l'adresse du nouveau segment
        mov ds, ax
```

```

        mov dx , offset new ; Charger l'offset de la nouvelle
adresse

        mov ax , 251Ch ; Charger le numéro de l'interruption
        int 21h
        ret
derout_1CH ENDP

; Procédure pour afficher 1 sec écoulée ...
affiche_1sec_ecoule PROC NEAR
    mov ax, seg data ; Charger le segment de données
    mov ds , ax
    mov dx , offset msg2 ; Charger l'adresse du message 2
    mov ah , 09h ; Afficher la chaîne de caractères
    int 21h
    ret
affiche_1sec_ecoule ENDP

; Procédure pour afficher 1 sec écoulée ...
affichemsg3 PROC NEAR
    mov ax, seg data ; Charger le segment de données
    mov ds , ax
    mov dx , offset msg3 ; Charger l'adresse du message 2
    mov ah , 09h ; Afficher la chaîne de caractères
    int 21h
    ret
affichemsg3 ENDP

; Procédure pour afficher les messages
affiche PROC NEAR
    mov ax, seg data
    mov ds , ax
    ;Affichage du msg1
    mov dx , offset msg1
    mov ah , 09h
    int 21h
    mov cx , 00003h
a:
    call NEAR PTR affiche_1sec_ecoule ; Appeler la procédure
pour afficher 1 sec écoulée ...
    loop a
    ; Affihcgae d'une nouvelle ligne
    mov dx , offset newline
    mov ah , 09h
    int 21h
    mov dx , offset newline

```

```

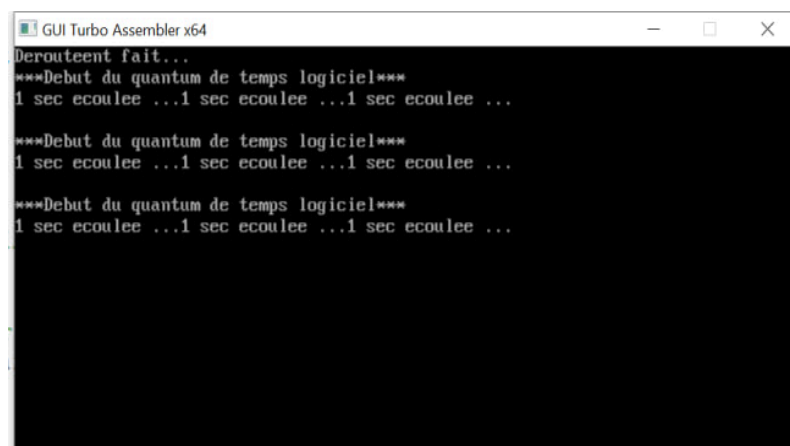
    mov ah , 09h
    int 21h
    ret
affiche endp
; La nouvelle routine d interruption pour int 1CH
new:
    mov ax , seg data
    mov ds , ax
    dec compt
    jnz fin ; Aller à fin si différent de zéro
    call NEAR PTR affiche ; Appeler la procédure pour afficher
les messages
    mov compt , 18 ; Réinitialiser le compteur
fin : iret

start:
    ; Initialisation de la pile
    mov ax , ma_pile
    mov ss , ax
    call NEAR PTR affichemsg3
    call NEAR PTR derout_1CH ; Appeler la procédure de
déroutement pour l'interruption 1CH

boucle:
    suite : jmp boucle ; Boucle infinie pour que le programme
continue à s'exécuter

code ENDS
END start

```



```

GUI Turbo Assembler x64
Derouteent fait...
***Debut du quantum de temps logiciel***
1 sec ecolee ...1 sec ecolee ...1 sec ecolee ...

***Debut du quantum de temps logiciel***
1 sec ecolee ...1 sec ecolee ...1 sec ecolee ...

***Debut du quantum de temps logiciel***
1 sec ecolee ...1 sec ecolee ...1 sec ecolee ...

```


4- Programme 2 :

```
data SEGMENT
    msg1 db "Tache 1 est en cours d'execution....", 0ah, 0dh, "$"
    msg2 db "Tache 2 est en cours d'execution....", 0ah, 0dh, "$"
    msg3 db "Tache 3 est en cours d'execution....", 0ah, 0dh, "$"
    msg4 db "Tache 4 est en cours d'execution....", 0ah, 0dh, "$"
    msg5 db "Tache 5 est en cours d'execution....", 0ah, 0dh, "$"
    newline db 13, 10, '$' ; Nouvelle ligne
    tacheCourante db 0 ; Variable pour suivre la tâche actuelle
    compt db 90 ; Compteur
data ENDS
ma_pile SEGMENT STACK
    dw 256 dup(?)
    tos label word
ma_pile ENDS
code SEGMENT
    assume CS:code, DS: data
; Définition de la procédure de déroutement pour l'interruption 1CH
    derout_1CH PROC NEAR
        derout:
            mov ax, seg new ; Charger l'adresse du nouveau segment
            mov ds, ax
            mov dx, offset new ; Charger l'offset de la nouvelle
adresse
            mov ax, 251CH ; Charger le numéro de l'interruption
            int 21H
            ret
    derout_1CH ENDP
    afficherMessage1 PROC NEAR
        mov ax, seg data ; Charger le segment de données
        mov ds, ax
        mov dx, offset msg1 ; Charger l'adresse du message 1
        mov ah, 09h ; Afficher la chaîne de caractères
        int 21h
        ret
    afficherMessage1 ENDP
    afficherMessage2 PROC NEAR
        mov ax, seg data
        mov ds, ax
        mov dx, offset msg2
        mov ah, 09h
        int 21h
        ret
    afficherMessage2 ENDP
    afficherMessage3 PROC NEAR
        mov ax, seg data
        mov ds, ax
        mov dx, offset msg3
```

```

    mov ah, 09h
    int 21h
    ret
afficherMessage3 ENDP
afficherMessage4 PROC NEAR
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg4
    mov ah, 09h
    int 21h
    ret
afficherMessage4 ENDP
afficherMessage5 PROC NEAR
    ; Initialiser DS avec l'adresse de la section de données
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg5
    mov ah, 09h
    int 21h
    ; Affihcer deux suate de ligne
    mov dx, offset newline
    mov ah, 09h
    int 21h
    mov dx, offset newline
    mov ah, 09h
    int 21h
    ret
afficherMessage5 ENDP
; Routine d interruption pour int 1CH
new:
    ; Charger DS avec l'adresse de la section de données
    mov ax, seg data
    mov ds, ax

    dec compt
    jnz fin
    ; incrementer le compteur de tâches
    inc tacheCourante
; Vérifier la tâche actuelle et appeler la procédure appropriée
    cmp tacheCourante, 1
    je tache1 ; Si egale affiche le msg1 sinon verifier la
suivante
    cmp tacheCourante, 2
    je tache2
    cmp tacheCourante, 3
    je tache3
    cmp tacheCourante, 4
    je tache4
    cmp tacheCourante, 5

```

```

        je tache5
tache1:
        ; Appelle a la procedure qui affiche le message
        call NEAR PTR afficherMessage1
        mov compt , 90 ; Initialiser le compteur
        jmp fin

tache2:
        call NEAR PTR afficherMessage2
        mov compt , 90
        jmp fin

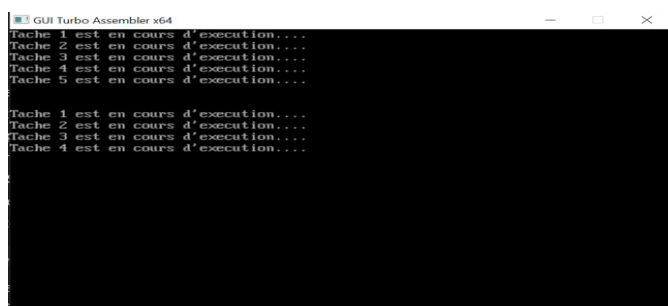
tache3:
        call NEAR PTR afficherMessage3
        mov compt , 90
        jmp fin

tache4:
        call NEAR PTR afficherMessage4
        mov compt , 90
        jmp fin

tache5:
        call NEAR PTR afficherMessage5
        mov compt , 90
        ;initialise le compteur des taches car on a arriver a la dernier
tache (5)
        mov tacheCourante, 0
        jmp fin
fin:
        iret

start:
        ; Configurer la pile
        mov ax, ma_pile
        mov ss, ax
        ; Appeler la procedure de deroutement
        call NEAR PTR derout_1CH
        ; Boucle infinie pour que le programme continue à s'exécuter
        boucle: jmp boucle
code ENDS
END start

```



```

GUI Turbo Assembler x64
Tache 1 est en cours d'execution...
Tache 2 est en cours d'execution...
Tache 3 est en cours d'execution...
Tache 4 est en cours d'execution...
Tache 5 est en cours d'execution...
Tache 1 est en cours d'execution...
Tache 2 est en cours d'execution...
Tache 3 est en cours d'execution...
Tache 4 est en cours d'execution...

```

5- Programme 1 avec modification :

```
data SEGMENT
    msg1 db  "***Debut du quantum de temps logiciel***", 0ah, 0dh, "$"
    msg2 db  "1 sec ecoulee ...$"
    msg3 db  "Derouteent fait...", 0ah, 0dh, "$"
    newline db 13, 10, '$'
    compt db 18 ; Compteur 1000/55 = 18.1
    secondes dw 0
data ENDS
ma_pile SEGMENT STACK
    dw 256 dup(?)
    tos label word
ma_pile ENDS
code SEGMENT
    assume CS:code, DS: data
; D?finition de la proc?dure de d?routement pour l'interruption 1CH
derout_1CH PROC NEAR
    derout:
        mov ax , seg new ; Charger l'adresse du nouveau segment
        mov ds , ax
        mov dx , offset new ; Charger l'offset de la nouvelle
adresse
        mov ax , 251Ch ; Charger le num?ro de l'interruption
        int 21h
        ret
derout_1CH ENDP

; Proc?dure pour afficher 1 sec ?coul?e ...
affiche_1sec_ecoule PROC NEAR
    mov ax, seg data ; Charger le segment de donn?es
    mov ds , ax
    mov dx , offset msg2 ; Charger l'adresse du message 2
    mov ah , 09h ; Afficher la cha?ne de caract?res
    int 21h
    ret
affiche_1sec_ecoule ENDP
; Proc?dure pour afficher 1 sec ?coul?e ...
affichemsg3 PROC NEAR
    mov ax, seg data ; Charger le segment de donn?es
    mov ds , ax
    mov dx , offset msg3 ; Charger l'adresse du message 2
    mov ah , 09h ; Afficher la cha?ne de caract?res
    int 21h
    ret
affichemsg3 ENDP
; Proc?dure pour afficher les messages
affiche PROC NEAR
    mov ax, seg data
    mov ds , ax
```

```

;Affichage du msg1
mov dx , offset msg1
mov ah , 09h
int 21h
mov cx , 00003h

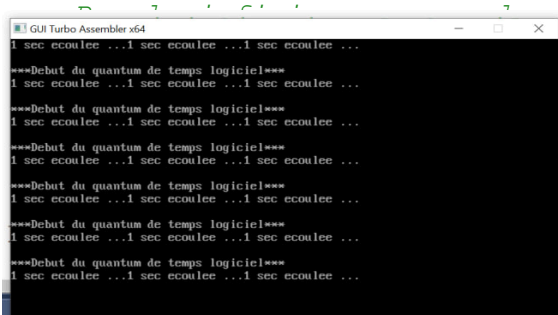
a:
call NEAR PTR affiche_1sec_ecoule ; Appeler la proc?dure pour
afficher 1 sec ?coul?e ...
loop a
; Affihcgae d'une nouvelle ligne
mov dx , offset newline
mov ah , 09h
int 21h
mov dx , offset newline
mov ah , 09h
int 21h
ret
affiche endp
; La nouvelle routine d interruption pour int 1CH
new:
mov ax , seg data
mov ds , ax
dec compt
jnz fin ; Aller ? fin si diff?rent de z?ro
call NEAR PTR affiche ; Appeler la proc?dure pour afficher les
messages
inc secondes
mov compt , 18 ; R?initialiser le compteur
fin : iret

start:
; Initialisation de la pile
mov ax , ma_pile
mov ss , ax
call NEAR PTR affichemsg3
call NEAR PTR derout_1CH ; Appeler la proc?dure de d?routement
pour l'interruption 1CH
mov ax , seg data
mov ds , ax

boucle:
cmp secondes , 300
jnz suite
mov ax , 4c00h
int 21h
suite : jmp boucle
continue ? s'ex?cuter

code ENDS
END start

```



6- Programme 1 avec modification :

```
data SEGMENT
    msg1 db "Tache 1 est en cours d'execution....", 0ah, 0dh, "$"
    msg2 db "Tache 2 est en cours d'execution....", 0ah, 0dh, "$"
    msg3 db "Tache 3 est en cours d'execution....", 0ah, 0dh, "$"
    msg4 db "Tache 4 est en cours d'execution....", 0ah, 0dh, "$"
    msg5 db "Tache 5 est en cours d'execution....", 0ah, 0dh, "$"
    newline db 13, 10, '$' ; Nouvelle ligne
    tacheCourante db 0 ; Variable pour suivre la t?che actuelle
    compt db 90 ; Compteur
    secondes dw 0 ; compteru de seconde
data ENDS

ma_pile SEGMENT STACK
    dw 256 dup(?)
    tos label word
ma_pile ENDS

code SEGMENT
    assume CS:code, DS: data

; D?finition de la proc?dure de d?routement pour l'interruption 1CH
derout_1CH PROC NEAR
    derout:
        mov ax, seg new ; Charger l'adresse du nouveau segment
        mov ds, ax
        mov dx, offset new ; Charger l'offset de la nouvelle adresse
        mov ax, 251CH ; Charger le num?ro de l'interruption
        int 21h
        ret
derout_1CH ENDP

afficherMessage1 PROC NEAR
    mov ax, seg data ; Charger le segment de donn?es
    mov ds, ax
    mov dx, offset msg1 ; Charger l'adresse du message 1
    mov ah, 09h ; Afficher la cha?ne de caract?res
    int 21h
    ret
afficherMessage1 ENDP

afficherMessage2 PROC NEAR
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg2
    mov ah, 09h
    int 21h
    ret
```

```
afficherMessage2 ENDP
```

```
afficherMessage3 PROC NEAR
```

```
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg3
    mov ah, 09h
    int 21h
    ret
```

```
afficherMessage3 ENDP
```

```
afficherMessage4 PROC NEAR
```

```
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg4
    mov ah, 09h
    int 21h
    ret
```

```
afficherMessage4 ENDP
```

```
afficherMessage5 PROC NEAR
```

```
    ; Initialiser DS avec l'adresse de la section de donn?es
    mov ax, seg data
    mov ds, ax
    mov dx, offset msg5
    mov ah, 09h
    int 21h
    ; Affihcer deux suate de ligne
    mov dx, offset newline
    mov ah, 09h
    int 21h
    mov dx, offset newline
    mov ah, 09h
    int 21h
    ret
```

```
afficherMessage5 ENDP
```

```
; Routine d interruption pour int 1CH
```

```
new:
```

```
    ; Charger DS avec l'adresse de la section de donn?es
    mov ax, seg data
    mov ds, ax
    dec compt
    jnz fin
    ; incrementer le compteur de t?ches
    inc tacheCourante
```

```
    ; V?rifier la t?che actuelle et appeler la proc?dure appropri?e
    cmp tacheCourante, 1
    je tache1 ; Si egale affiche le msg1 sinon verifier la suivante
```

```

cmp tacheCourante, 2
je tache2
cmp tacheCourante, 3
je tache3
cmp tacheCourante, 4
je tache4
cmp tacheCourante, 5
je tache5

```

```

tache1:
; Appelle a la procedure qui affiche le message
call NEAR PTR afficherMessage1
inc secondes
mov compt , 90 ; Initialiser le compteur
jmp fin

```

```

tache2:
call NEAR PTR afficherMessage2
inc secondes
mov compt , 90
jmp fin

```

```

tache3:
call NEAR PTR afficherMessage3
inc secondes
mov compt , 90
jmp fin

```

```

tache4:
call NEAR PTR afficherMessage4
mov compt , 90
jmp fin

```

```

tache5:
call NEAR PTR afficherMessage5
inc secondes
mov compt , 90
;initialise le compteur des taches car on a arriver a la dernier tache

```

(5)

```

mov tacheCourante, 0
jmp fin

```

```

fin:
iret

```

```

start:
; Configurer la pile
mov ax, ma_pile
mov ss, ax

; Appeler la procedure de routement

```



```
call NEAR PTR derout_1CH
```

```
; Boucle infinie pour que le programme continue ? s'ex?cuter
```

```
boucle:
```

```
mov ax, seg data
```

```
mov ds, ax
```

```
cmp secondes , 60 ; Comme l'affichage se fait chaque 5 secondes donc  
apres 60 affichage en s arrete car 60*5 = 300 sec (5min)
```

```
jnz suite
```

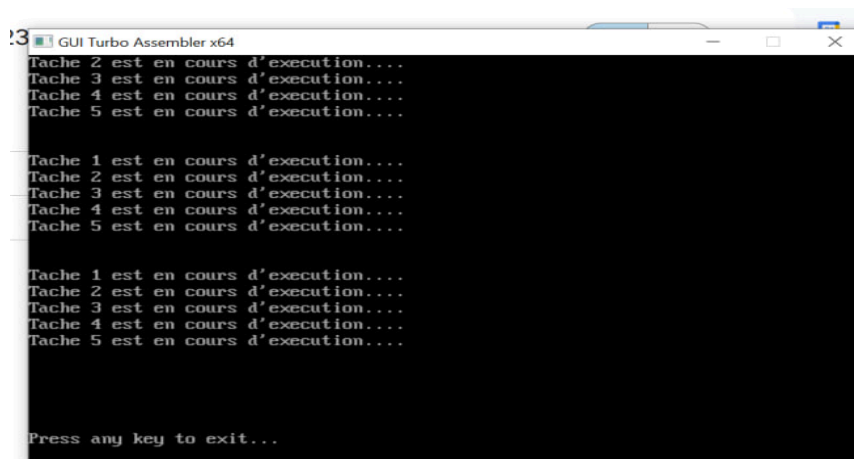
```
mov ax, 4c00h
```

```
int 21h
```

```
suite : jmp boucle
```

```
code ENDS
```

```
END start
```



```
GUI Turbo Assembler x64
Tache 2 est en cours d'execution....
Tache 3 est en cours d'execution....
Tache 4 est en cours d'execution....
Tache 5 est en cours d'execution....

Tache 1 est en cours d'execution....
Tache 2 est en cours d'execution....
Tache 3 est en cours d'execution....
Tache 4 est en cours d'execution....
Tache 5 est en cours d'execution....

Tache 1 est en cours d'execution....
Tache 2 est en cours d'execution....
Tache 3 est en cours d'execution....
Tache 4 est en cours d'execution....
Tache 5 est en cours d'execution....

Press any key to exit...
```

Merci Pour votre attention