

Architecture (ASR1 et ASR2)

TP N° 7

Objectifs :

- Comprendre les mécanismes d'interruption
- Déclenchement d'une Interruption logicielle IT DOS : codage en assembleur

1 Organisation du TP :

Le travail effectué durant cette séance sera à rendre à votre responsable de TP par email impérativement à la fin de la séance. Le sujet de l'email sera [2012][Architecture][TP7] Nom Prénom .

Le TP comporte plus d'exercices que vous ne pourrez normalement en faire pendant la séance. Ces exercices supplémentaires-vous sont donnés à titre d'entraînement et votre responsable de TP pourra vous les corriger si vous les lui rendez. Attention, corriger vos exercices ne veut pas dire qu'il vous fournira une correction type des exercices ! Il vous indiquera des pistes pour résoudre vos problèmes et pour vous aider à trouver une solution correcte qui fonctionne.

2 Complément de cours sur les interruptions

2.1 Les principales interruptions du BIOS

Les IT de 0 à 7 sont appelées directement par l'U.C. du microprocesseur. Il s'agit de :

- IT 0 : division par 0
- IT 1 : pas à pas
- IT 2 : circuit RAM défectueux
- IT 3 : point d'arrêt
- IT 4 : débordement
- IT 5 : copie

Les IT suivantes peuvent être appelée normalement :

- IT 8 : tic horloge (timer) appelé toutes les 55 ms (18,2 fois/s.). Elle servait par exemple à arrêter le moteur de l'unité de disquette lorsqu'aucun accès à la disquette n'était exécuté. Cette interruption 08H, après avoir lancé le programme correspondant, appelle l'interruption 1CH. Cette dernière ne contient qu'un retour IRET afin de permettre aux programmeurs d'implanter leurs propres programmes d'interruption (objectifs: programmation concurrente, multitâche,...)
- IT 10h à 12h : variable selon le matériel connecté. Souvent, IT 10 désigne l'IT vidéo IT 11 fournit la liste du matériel disponible et IT 12 le calcul de l'espace mémoire disponible
- IT 13h : gestion des disques
- IT 14h : gestion de la porte série
- IT 15h : gestion d'un lecteur de cassettes
- IT 16h : gestion du clavier
- IT 17h : imprimante
- IT 18h : activation du BIOS
- IT 19h : reset du système
- IT 1Ah : gestion de la date et de l'heure

Les routines associées au DOS travaillent à un niveau supérieur. Elles peuvent parfois faire appel à des routines du BIOS. Les principales interruptions associées au DOS sont :

- IT 20h : fin normale d'un programme
- IT 21h : appel de sous-routines système

Architecture (ASR1 et ASR2)

TP N° 7

- **IT 22h** : gestion de l'adresse de saut après l'exécution d'un programme. Elle ne peut être activée directement. Elle précise la routine qui reprend le contrôle après la fin du programme
- **IT 23h** : gestion de l'adresse de saut après un control-break. Idem à ci-dessus.
- **IT 24h** : gestion de l'adresse de saut en cas d'erreur
- **IT 25h** : lecture directe sur disque
- **IT 26h** : écriture directe sur disque
- **IT 27h** : programme résident (terminer un programme en le laissant résident). Un programme est dit résident si, une fois son exécution terminée, il rend le contrôle à l'environnement appelant mais reste chargé en mémoire. Il pourra alors être réactivé à tout moment, notamment à l'aide d'une IT.

2.2 Interruptions et Systèmes d'exploitation

De nos jours, les systèmes d'exploitation prennent eux même en charge les interruptions. Ils ne laissent plus aux programmes la possibilité de piloter directement celles-ci et proposent un système de signaux ou d'événements à la place.

Nous ne regarderons pas ces mécanismes qui devraient être abordés dans le cours « Système 2 » et nous nous concentrerons sur le mécanisme standard des interruptions (que nous pouvons encore utiliser via le langage assembleur).

3 Exercice 1 : Déclenchement d'une interruption logicielle (Obligatoire)

Lorsqu'une IT est déclenchée le microprocesseur :

- empile les indicateurs du registre d'état
- désactive les IT matérielles
- empile CS
- empile IP

Il exécute ensuite la routine d'interruption. Cette routine se termine par un IRET qui a pour effet de :

- dépiler IP
- dépiler CS
- dépiler les indicateurs

Pour déclencher une interruption en assembleur il suffit d'invoquer le numéro de l'interruption par l'instruction INT.

3.1 Affichage de l'heure avec procédure interne :

Vous allez invoquer dans une boucle, l'interruption qui donne l'heure courante de votre PC et vous afficherez ce résultat avec le format suivant :

```
L heure actuelle est : 12h01m36s11
L heure actuelle est : 12h01m36s43
L heure actuelle est : 12h01m37s15
```

Rappel : l'interruption de l'horloge est INT 21h / AH=2Ch Des détails sont disponibles dans l'aide de emu8086.

Vous utiliserez la procédure suivante pour convertir les valeurs retournées par l'interruption en des valeurs affichables :

```
; Converti le premier paramètre (valeur entre 0 et 99) en 2 caractères ASCII
; IN   : paramètre 1 : la valeur à convertir en texte
;      : paramètre 2 : vide, il sert juste pour le retour
; OUT  : paramètre 1 : la valeur ASCII de l'unité
;      : paramètre 2 : la valeur ASCII de la dizaine
```

Architecture (ASR1 et ASR2)

TP N° 7

```

conversionValeurASCII PROC NEAR
    PUSH BP
    PUSH AX
    PUSH BX

    MOV BP, SP
    MOV AX, [BP+10]

    MOV BL, 10
    DIV BL

    MOV BL, AH
    MOV BH, 0
    ADD BX, 48

    MOV [BP+10], BX

    MOV AH, 0
    ADD AX, 48

    MOV [BP+8], AX

    POP BX
    POP AX
    POP BP

    RET
conversionValeurASCII ENDP

```

Quelle est la durée moyenne d'exécution d'un passage dans la boucle ?

3.2 Affichage de l'heure avec procédure externe :

Modifiez votre programme précédent pour qu'il utilise la procédure `conversionValeurASCII` comme une procédure externe. Vous devrez également créer un fichier `.obj` contenant cette procédure à l'aide de `masm`. Faites de même avec vos propres procédures.

4 Exercice 2 : Gestion de l'affichage (conseillé)

4.1 Affichage du texte simplifié

Modifier le programme fait dans l'exercice précédent pour qu'il affiche les modifications de l'heure sur la ligne précédemment utilisée de manière à ce que seule le texte après les `' : '` soit mis à jour.

4.2 Affichage du texte en couleur

Modifier le programme fait dans l'exercice précédent pour qu'il affiche l'heure en couleur. La couleur devra changer à chaque affichage.

5 Exercice 3 : Procédures d'interruptions (Obligatoire)

Jusqu'à maintenant, nous avons déjà vu comment utiliser des interruptions déjà défini. Mais comment faire pour écrire son propre traitant d'interruption ? C'est ce que nous allons voir maintenant.

5.1 Ecrire une interruption avec les fonctions DOS

Parmi l'ensemble des interruptions `21h`, nous en avons plusieurs qui nous permettent de gérer le vecteur d'interruption.

Architecture (ASR1 et ASR2)

TP N° 7

5.1.1 INT 21h, AH=35h

Cette interruption prend en paramètre, via AL, le numéro d'une interruption et nous donne, pour celle-ci, les adresses contenues dans le vecteur d'interruption. BX contient l'adresse du traitant et ES contient son numéro de segment.

5.1.2 INT 21h, AH=25h

Cette interruption prend trois paramètres. Dans AL on retrouve le numéro de l'interruption que l'on souhaite modifier, dans DX l'adresse du traitant et dans DS l'adresse du segment où se trouve le traitant.

5.1.3 A vous de jouer !

Les deux dernières interruptions, que nous avons vues, fonctionnent sous emu8086. Il est conseillé de travailler dans ce dernier pour bénéficier de son environnement de travail et éviter les modifications abusives du vecteur d'interruption.

5.1.3.1 Etape 1 :

Nous allons commencer par écrire un petit programme tout simple qui affiche « bonjour » à l'aide d'une procédure que nous nommerons « TraitantInt ». La procédure prendra un paramètre (l'adresse de « bonjour ») via le registre DX.

5.1.3.2 Etape 2 :

Nous allons changer cette procédure en traitant d'interruption.

5.1.3.3 Etape 3 :

Parmi l'ensemble des interruptions, nous allons modifier l'interruption FDh existante pour la remplacer par la nôtre. Complétez le programme suivant :

```
STACK SEGMENT STACK
    DW 256 DUP (0)
base:
STACK ENDS

DATA SEGMENT
    msg DB "Ma première interruption",10,13,'$'
    ancien DW ?,?
    nouveau DW ?,?
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK

TraitantInt PROC
    ;; à compléter
TraitantInt ENDP

debut:
    MOV AX, Data
    MOV DS, AX
    MOV AX, Stack
    MOV SS, AX
    MOV SP, base

; Sauvegarde de l'ancien traitant
```

Architecture (ASR1 et ASR2)

TP N° 7

```

;; à compléter

; Sauvegarde de DS
;; à compléter

; Préparation des paramètres pour la mise en place du traitant
;; à compléter

; Mise à jour du vecteur d'interruption
;; à compléter

; Restauration de DS
;; à compléter

; Appel de l'interruption
;; à compléter

; Restauration du vecteur d'interruption
;; à compléter

fin:
    MOV AH, 4CH
    INT 21H

CODE ENDS
END debut
  
```

Une fois que votre programme fonctionne avec emu8086, testez-le avec masn.

5.2 Ecrire sa première interruption à la main

Ici nous devrions utiliser masn car emu8086 ne nous permet pas d'écrire dans la zone mémoire correspondant au vecteur d'interruptions sans passer par les interruptions. **Attention, ne pas exécuter le programme avant la fin sans quoi vous pouvez endommager votre vecteur d'interruption.**

5.2.1 Etape 1 : écriture d'un programme simple

Nous allons commencer par écrire un petit programme tout simple qui affiche « Hello » à l'aide d'une procédure externe que nous nommerons « TraitantInt ». La procédure prendra un paramètre (l'adresse de « hello ») via le registre DX.

Testez votre programme avant de passer à la suite.

5.2.2 Etape 2 : installation du traitant d'interruption dans le vecteur d'interruption

Attention, nous allons, durant cette étape, modifier le vecteur d'interruption. C'est-à-dire que nous allons modifier la zone mémoire allant de l'adresse 0000h à l'adresse 0400h.

Parmi l'ensemble des interruptions, nous allons modifier l'interruption FDh existante pour la remplacer par la nôtre. Dans le vecteur d'interruption, chaque interruption occupe 4 octets. Les deux premiers pour l'adresse de l'interruption dans son segment. Les deux derniers pour l'adresse du segment contenant l'interruption.

Dans notre cas, l'adresse de l'interruption correspond à l'adresse de l'étiquette de la procédure. Celle-ci étant dans le segment de code, l'adresse du segment correspond à celle du registre CS.

Mais où doit-on écrire ces adresses ? Comme nous l'avons déjà vu, chaque interruption occupent 4 octets dans le vecteur d'interruption. Or nous voulons modifier l'interruption FDh. Celle-ci se trouve donc à l'adresse 0000h

Architecture (ASR1 et ASR2)

TP N° 7

(début du vecteur d'interruption) + $FDh \times 4$ (chaque interruption occupant 4 octets). Notre interruption est donc définie à l'adresse 03F4h.

Commençons par positionner le segment de données (segment dans lequel on sait facilement écrire) à l'adresse 0000h. On va sauver la valeur initiale du registre DS pour la restaurer une fois notre traitant d'interruption installé :

```
MOV AX, DS      ;; sauvegarde de DS
MOV ES, AX
MOV AX, 0000h   ;; DS pointe sur le début de la mémoire
MOV DS, AX
```

Il faut maintenant placer l'adresse de la procédure d'interruption en 03F4h, comme suit :

```
MOV BX, 03F4h
MOV [BX], TraintantInt
```

Et, l'adresse du segment de code dans les 2 derniers octets :

```
MOV BX, 03F6h
MOV AX, CS
MOV [BX], AX
```

Pensez à restaurer la valeur initiale de DS, sans quoi votre programme ne fonctionnera pas.

5.2.3 Etape 3 : Transformation de la procédure en interruption

Il n'y a pas de réelle différence entre une procédure externe classique et une procédure d'interruption sauf que cette dernière se termine par un IRET et non un RET et qu'on utilise l'instruction INT et non CALL pour l'appeler.

Commencez par modifier votre procédure pour qu'elle corresponde à une procédure d'interruption : supprimer le mot clef FAR et changez RET en IRET.

Changez votre appel à la procédure par un appel à l'interruption FDh que nous venons d'installer.

5.2.4 Etape 4 : sauvegarde et restauration

Ecrivez un morceau de code qui sauvera dans deux variables de type double (« oldadd » et « oldseg ») les valeurs présentes dans le vecteur d'interruption avant la modification.

Ecrivez un autre morceau de code qui restaurera les valeurs sauvées avant la fin de votre programme.

Testez votre programme après l'avoir compiler avec masm. Attention, si vous avez mal fait la partie sur la sauvegarde et la restauration des valeurs vous allez endommager le vecteur d'interruption de la machine.

6 Exercice 4 : Passage en mode graphique (Conseillé)

Nous allons voir maintenant comment passer en mode graphique et faire des dessins.

Dans un premier temps, vous allez devoir passer un mode graphique. Pour cela on utilisera l'interruption INT 10h / AH = 0 et vous spécifierez via AL le mode vidéo désiré. On utilisera le mode 13h pour avoir un mode graphique correct (bien que minimal : 256 couleurs. 320x200 pixels). Pour profiter véritablement du mode graphique, vous devez activer le mode graphique en sélectionnant la page 1 (INT 10h / AH = 05h).

Pour afficher un point nous allons utiliser l'interruption INT 10h / AH = 0Ch.

Architecture (ASR1 et ASR2)

TP N° 7

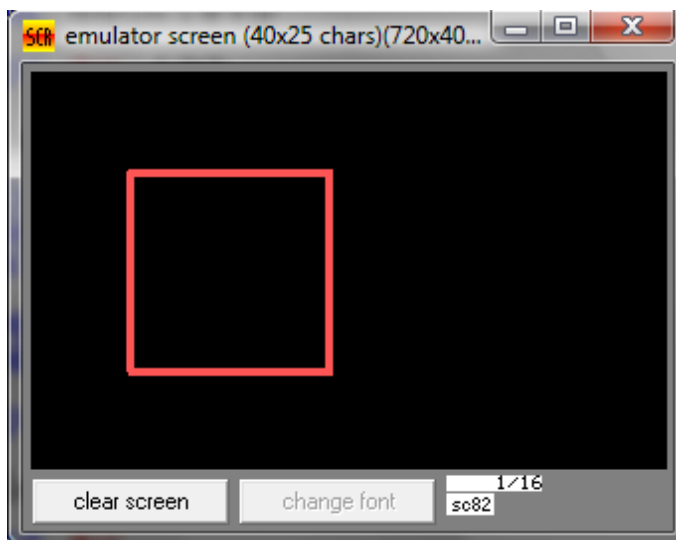
6.1 Premier test

Réalisez un programme simple (sans procédure) qui affiche un carré de 100 sur 100 pixels comme sur la capture d'écran ci-dessous.

6.2 Fonctions

Modifiez le programme précédent pour qu'il utilise 4 procédures dont les paramètres seront passés par la pile. Voici la spécification d'une des procédures :

```
;affiche une ligne vers la droite
;IN : P1 = colonne origine
;   : P2 = ligne origine
;   : P3 = couleur
;   : P4 = taille en pixels
;OUT: aucun
```



6.3 Bibliothèque

Externalisez les fonctions dans un autre fichier que celui de votre application. Et dessinez une autre figure !

7 Exercice 5 : lecture et écriture de fichiers (conseillé)

7.1 Création d'un répertoire

A l'aide de l'interruption INT 21h/39h, vous allez créer un programme contenant 2 procédures :

- La première devra lire au clavier un chemin d'accès de la forme « c:\ASR2\TP6\\$. ». Le \$ servant à indiquer la fin de la lecture.
- La deuxième, devra créer le répertoire correspondant à ce chemin

7.2 Création d'un fichier

Ajouter une procédure à votre programme précédent qui crée un fichier (INT 21h / AH= 3Ch) dont le nom est fourni pas une variable *filename*.

Vous modifierez votre programme pour que maintenant il crée aussi le fichier dans le répertoire spécifié.

7.3 Lecture d'un fichier

Pour pouvoir lire un fichier, il est nécessaire de suivre l'ordre des opérations suivantes :

- Ouverture du fichier (INT 21h / AH= 3Dh)
- Lecture du fichier (INT 21h / AH= 3Fh)
- Fermeture du fichier (INT 21h / AH= 3Eh)

Ajouter une procédure à votre programme qui effectue ces trois opérations.

Le fichier à ouvrir sera spécifié par une variable *dir* contenant le chemin d'accès et une variable *filename* contenant le nom du fichier.

Le résultat sera stocké dans une variable de type tableau créée en mémoire.

7.4 Ecriture d'un fichier

Pour pouvoir écrire un fichier, il est nécessaire de suivre l'ordre des opérations suivantes :

- I. Ouverture du fichier (INT 21h / AH= 3Dh)
- II. Ecriture du fichier (INT 21h / AH= 40h)
- III. Fermeture du fichier (INT 21h / AH= 3Eh)

Architecture (ASR1 et ASR2)

TP N° 7

Ajouter une procédure à votre programme qui effectue ces trois opérations.

Le fichier à ouvrir sera spécifié par une variable *dir* contenant le chemin d'accès et une variable *filename* contenant le nom du fichier.

Les données à écrire seront également en mémoire et définies par une variable *data2write*.

8 Exercice 6 : Bibliothèque de lecture et écriture de fichiers (Optionnel)

8.1 Création d'une bibliothèque *fileio.obj*

Modifier votre programme précédent pour que celui ne contienne plus que des procédures et que celles-ci soient appelables de l'extérieur du programme. Créer un fichier *.obj* de cette bibliothèque.

8.2 Réalisation d'un programme utilisant cette bibliothèque

Ecrire un programme simple, qui ouvre un fichier *TOTO.txt* (contenant du texte sans accent) et en copie le contenu dans un fichier *TITI.txt* après avoir transformé tous les o en i.

Exemple de fichier *TOTO.txt*

*This software is intended for academic, commercial and personal purposes.
The license is not limited to any number of workstations.*

Exemple de fichier *TOTO.txt*

*This sifware is intended fir academic, cimmercial and persinal purpises.
The license is nit limited to any number if wirkstatiins*

9 Exercice 7 : Procédures d'interruptions en C (Optionnel)

Les programmes d'interruptions peuvent être codés en C également. Ici nous voulons réaliser un affichage toutes les 50 ms en haut à droite de l'écran.

Le vecteur *ox1C* correspond à l'interruption matérielle cyclique dont la fréquence 1/18ème de seconde (50 ms). La procédure *Affiche* est appelée chaque 50 ms : elle affiche un message dans le coin supérieur droit de l'écran chaque seconde.

Attention : On va utiliser *Borlandc* pour compiler ce fichier C car *visual C* ne permet pas l'utilisation des fonctions d'interruptions.

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>

void Ecr(char c) {
    int j=0;
    int k=0;
    while(c[j]!='\0') {
        pokeb(0xB800, 0x0090+k,c[j]);
        j++;
        k+=2;
    }
}
```

Architecture (ASR1 et ASR2)

TP N° 7

```
} /*Attention : oxBooo pour un écran Noir et Blanc */  
}  
void interrupt Affiche() {  
    static int cpt = 0;  
    static int per = 18;  
    cpt++;  
    if (cpt == (per-5))  
        Ecr("CYCLIC");  
    if (cpt == per) {  
        Ecr("  ");  
        cpt = 0;  
    }  
}  
void main() {  
    char c;  
    void interrupt (*old) ();  
    /* TODO  
     * Installation de la procedure d'IT  
     * (oxiC est associe a Affiche)  
     */  
    clrscr();  
    /* Récupération de l'adresse de la procedure d'IT oxiCh*/  
    printf("\nHorloge tourne : Taper 'o' pour sortir") ;  
  
    /* TODO  
     * Boucle qui teste le caractère o  
     */  
  
    /* TODO  
     * Restauration de la procedure d'IT  
     */  
  
    printf ("Fini : vecteur initial réinitialisé");  
}
```