Chapitre 10

Les interruptions logicielles

Nous avons rencontré la notion de sous-programme et nous avons vu comment mettre en place un tel sous-programme. Un sous-programme peut être conçu par l'utilisateur. Certains sous-programmes importants peuvent être conçus (avec beaucoup de soins) pour être réutilisés par un grand nombre d'utilisateurs. En général ils sont placés dans des *bibliothèques* ; nous verrons plus tard comment. Il est également prévu qu'un certain nombre d'entre eux, les plus importants, puissent être appelés par une méthode quelque peu spécifique : ce sont les **routines de service** associées à des *interruptions logicielles*. Rappelons que **routine** est un synonyme de sous-programme.

Comme leur nom l'indique, les *interruptions* (*interrupt* en anglais) viennent interrompre le déroulement normal du microprocesseur. Elles peuvent être générées par le matériel (par exemple à cause d'une division par zéro), par appel d'un périphérique (demande du clavier) ou par une instruction spécifique du langage machine. On parle dans ce dernier cas d'**interruption logicielle**.

10.1 Appel des interruptions

10.1.1 Syntaxe

Syntaxe.- La syntaxe de l'appel d'une interruption logicielle en langage symbolique pour le microprocesseur `i8086` est tout simplement :

```
int xx
```

où `xx` est un entier compris entre `00h` et `FFh`.

Il y a donc 256 interruptions possibles pour le microprocesseur `i8086`.

Interruptions implémentées.- Intel s'est réservé un certain nombre d'interruptions (par exemple liée à la division par zéro), sans leur associer de routines de service. C'est aux concepteurs du BIOS et du système d'exploitation d'implémenter les routines de service des interruptions réservées et de concevoir les autres.

Il n'existe pas en général 256 interruptions implémentées ; leur nombre dépend du système d'exploitation et de ce qui a été prévu par les grands logiciels utilisateur.

Passage des paramètres.- Certaines interruptions ont besoin de paramètres (en nombre limité). Le nombre de paramètres dépend de la routine de service associée à l'interruption. Rien n'est prévu pour le passage des paramètres. On se sert donc des variables globales que sont les contenus des registres, à la fois pour les paramètres d'entrée et les paramètres de sortie. En complément, on se sert de la pile si le nombre de registres est insuffisant.

10.1.2 Un exemple

Introduction.- Nous avons vu le principe de mise en place des entrées-sorties en utilisant les instructions `in` et `out`. Nous ne sommes pas allés très loin pour l'instant, vu la délicatesse de la mise en œuvre. L'interruption `21h` du système d'exploitation DOS nous permet d'accéder plus facilement aux entrées-sorties en implémentant les gestionnaires sous forme d'interruption logicielle.

Nous allons utiliser cette façon d'accéder aux entrées-sorties sans voir, dans une première étape, comment les routines de service associées sont implémentées.

Fonctions d'une interruption.- Puisqu'il n'y a que 256 interruptions possibles et que les concepteurs du système d'exploitation MS-DOS avaient envie d'en implémenter plus, ils ont associés plusieurs actions à une même interruption logicielle. Chaque cation s'appelle une **fonction** de l'interruption et se distingue des autres actions par la valeur du registre `AH`, appelée **numéro de la fonction**. Par exemple la fonction (de numéro) `02h` de l'interruption `21h` permet d'afficher un caractère à l'écran.

Affichage grâce au DOS.- Pour afficher un caractère à l'écran grâce à la fonction `02h` de l'interruption `21h` du DOS, on place la valeur de ce caractère, c'est-à-dire son code ASCII, dans le registre `DL`.

Exemple.- Écrivons un programme permettant d'afficher la lettre 'a' à l'écran.

Il faut, pour cela, savoir que le code ASCII de 'a' est `61h`.

```
C:>debug
-a
249C:0100 mov dl,61
249C:0102 mov ah,02
249C:0104 int 21
249C:0106
-g
a
Le programme a provoqu\'e une erreur de d\'epassement de division.
Si le probl\'eme persiste consultez votre fournisseur.
C:>
```

qui nous donne le résultat attendu (l'affichage de 'a') mais qui provoque également une erreur.

10.1.3 Retour au système d'exploitation

Introduction.- L'erreur provoquée par le programme précédent est due au fait qu'on ne lui dit pas ce qu'il faut faire après qu'il ait effectué l'affichaque de 'a'.

Lorsqu'on termine un programme lancé par un système d'exploitation, on a évidemment envie de revenir à ce système d'exploitation, par exemple pour pouvoir lancer un autre programme. Ceci n'est évidemment pas prévu par le microprocesseur, qui ne peut pas savoir comment sera structuré le système d'exploitation. Le concepteur du système d'exploitation doit donc prévoir un tel retour. Le sous-programme associé est en général implémenté sous forme d'une interruption logicielle, vu son importance.

Retour au DOS.- Le retour au système d'exploitation DOS se fait grâce à la fonction `4C00h` de l'interruption `21h`, à placer dans le registre `ax`. Une version propre du programme précédent est donc :

```
C:>debug
-a
249C:0100 mov dl,61
249C:0102 mov ah,02
249C:0104 int 21
249C:0106 mov ax,4c00
249C:0109 int 21
249C:010B
-g
a
C:>
```

Remarques.- 1^o) En fait il s'agit de la fonction `4Ch` de l'interruption `21h`. On place éventuellement un code d'erreur dans le registre `al`, ce qui permet des tests dans un fichier batch. Par défaut on y place `00h` et on abrège :

```
mov ah,4C
mov al,00
int 21h
```

en :

```
mov ax,4C00h
int 21h
```

2^o) On peut s'étonner qu'une fonction aussi importante porte un numéro si élevé. En fait, dans le première version du DOS, il s'agissait de l'interruption `int 20h`. Puis il s'est agi de la fonction `00h` de l'interruption `int 21h`. Ces deux versions sont conservées pour des raisons de compatibilité ascendante

mais elles sont remplacées, comme nous venons de le voir, par la fonction `4Ch` de l'interruption `int 21h`.

10.1.4 Trace d'un programme avec interruption

Introduction.- Nous avons vu comment effectuer la trace d'un programme avec le logiciel `debug`. Il y a un problème lorsqu'un programme contient des interruptions (logicielles). En effet si on utilise la commande `T`, lorsqu'on arrive à l'interruption, on va décrire toutes les instructions de celle-ci. Ce n'est pas ce qui est voulu en général ; on suppose que le concepteur de l'interruption a bien mis en place la routine de service associée. On préférerait donc une commande qui exécute la routine de service en son entier et nous montre le résultat après celle-ci.

La commande P.- Une telle commande existe pour `debug`, il s'agit de la commande `P` (pour l'anglais *to Proceed*).

Exemple.- Déterminons la capacité de la mémoire vive de notre ordinateur.

En ce qui concerne les compatibles PC, celle-ci était déterminée au moment du démarrage de l'ordinateur et le résultat placé à l'emplacement de la mémoire vive (réservé par le BIOS) `0040:0013h`.

Il existe une interruption du BIOS, l'interruption logicielle `int 12h`, qui permet de récupérer cette valeur et de la placer dans le registre `ax`.

Concevons donc un programme permettant de déterminer la taille de la mémoire :

```
C:\>debug
-a
17A4:0100 int 12
17A4:0102 nop
17A4:0103
```

Exécutons maintenant la trace de ce programme :

```
C:\>debug
-a
17A4:0100 int 12
17A4:0102 nop
17A4:0103
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0100   NV UP EI PL NZ NA PO NC
17A4:0100 CD12          INT     12
-P

AX=0280  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0102   NV UP EI PL NZ NA PO NC
17A4:0102 90            NOP
-
```

qui nous donne la valeur. Celle-ci doit de nos jours être interprétée car nous dépassons la capacité mémoire prévue lors de la conceptions des premières versions du BIOS.

Remarque.- Si on effectue la trace de ce programme en utilisant la commande `T` au lieu de la commande `P`, on obtient :

```
C:\>debug
-a
17A4:0100 int 12
17A4:0102 nop
17A4:0103
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0100   NV UP EI PL NZ NA PO NC
17A4:0100 CD12          INT     12
-t

AX=0000  BX=0000  CX=0000  DX=0000  SP=FFE8  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=F000  IP=F841   NV UP DI PL NZ NA PO NC
F000:F841 1E            PUSH    DS
-t

AX=0000  BX=0000  CX=0000  DX=0000  SP=FFE6  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=F000  IP=F842   NV UP DI PL NZ NA PO NC
F000:F842 B84000        MOV     AX,0040
-t

AX=0040  BX=0000  CX=0000  DX=0000  SP=FFE6  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=F000  IP=F845   NV UP DI PL NZ NA PO NC
F000:F845 8ED8          MOV     DS,AX
-t

AX=0040  BX=0000  CX=0000  DX=0000  SP=FFE6  BP=0000  SI=0000  DI=0000
DS=0040  ES=17A4  SS=17A4  CS=F000  IP=F847   NV UP DI PL NZ NA PO NC
F000:F847 A11300        MOV     AX,[0013]                       DS:0013=0280
-t

AX=0280  BX=0000  CX=0000  DX=0000  SP=FFE6  BP=0000  SI=0000  DI=0000
DS=0040  ES=17A4  SS=17A4  CS=F000  IP=F84A   NV UP DI PL NZ NA PO NC
F000:F84A 1F            POP     DS
-t

AX=0280  BX=0000  CX=0000  DX=0000  SP=FFE8  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=F000  IP=F84B   NV UP DI PL NZ NA PO NC
F000:F84B CF            IRET
-t

AX=0280  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17A4  CS=17A4  IP=0102   NV UP EI PL NZ NA PO NC
17A4:0102 90            NOP
-
```

qui entre dans le détail de la routine de service de l'interruption.

Ici celle-ci est simple : elle consiste à aller chercher le renseignement là où il est placé. Mais en général c'est beaucoup plus compliqué et cela ne nous intéresse pas, sauf pour aller voir comment est conçue cette routine.

10.2 Fonctionnement des interruptions

Effet d'une interruption.- Lorsqu'une interruption est exécutée, le microprocesseur sauvegarde automatiquement le registre des indicateurs, le pointeur d'instruction `IP`, le registre du segment de code `CS` sur la pile et il exécute les instructions à partir de l'instruction située à un emplacement mémoire déterminé.

Tableau des vecteurs d'interruption.- Quelle est l'adresse de cette instruction ? Elle est donnée par le **tableau des vecteurs d'interruption** (en anglais *interrupt vector table*). L'emplacement mémoire de cette instruction est donné par les quatre octets successifs dont le premier est à l'adresse le quadruple du numéro de l'instruction.

Par exemple l'adresse de `INT 03` se trouve à partir de `000Ch` puisque :

$$4 \times 3 = 12 = 0Ch.$$

Comment décoder l'adresse ? Les deux premiers octets déterminent la valeur de `IP` et les deux derniers la valeur de `CS`. On obtient ainsi l'adresse logique `CS:IP`. N'oublions pas que le premier octet correspond aux huit bits de poids faible et le deuxième aux huit bits de poids fort.

Les 1024 premiers bits de la mémoire vive d'un ordinateur dont le microprocesseur est un `i8086` [1] sont donc occupés par le tableau des vecteurs d'interruption.

Exemple.- Les adresses ne sont pas imposées par les concepteurs du `i8086` mais par les concepteurs des routines de service associées. Elles dépendent donc de la version du BIOS, de la version du DOS et des programmes utilisateur suivant que l'interruption est implémentée par les concepteurs du BIOS, du système d'exploitation ou du programme utilisateur.

On peut se servir de `debug` pour obtenir les adresses de notre système. Par exemple pour l'interruption `INT 03`, on obtient sur mon système :

```
C:>debug
-d 0000:000C L4
0000:0000 65 04 70 00                    e.p.
-q
```

ce qui donne l'adresse `0070:0465`.

Routine de service d'une interruption.- Le sous-programme commençant à cette adresse s'appelle la **routine de service de l'interruption** (en anglais *interrupt service routine*, abrégé en `ISR`, ou *interrupt handler*).

Le sous-programme doit se terminer par **IRET** (pour *interrupt return*, et non plus par `RET`), ce qui permet de récupérer automatiquement, non seulement `CS` et `IP` mais également le registre des indicateurs.

1. Ainsi que pour le mode réel pour les microprocesseurs Intel suivants. C'est différent pour le mode dit protégé.

10.3 Conception des interruptions logicielles

Nous avons vu ci-dessus, lors de la trace de notre programme contenant `int 12h`, comment l'interruption logicielle `int 12h` était implémentée.

10.4 Types d'interruptions

On distingue trois types d'interruptions d'après la façon dont elles sont appelées : les interruptions appelées par la microprocesseur lui-même, celles appelées par le logiciel et celles appelées par les périphériques. On parle quelquefois d'**exceptions** pour les deux premiers types.

Interruptions appelées par le microprocesseur.- Les concepteurs du `i8086` ont réservé les interruptions de numéro `00H` à `12h` pour être appelées par le microprocesseur. On parle quelquefois d'**interruptions internes**. Ces interruptions sont appelées par le microprocesseur en général en réponse à une erreur, par exemple l'interruption `00h` lorsqu'on essaie de diviser par 0.

C'est à l'utilisateur du `i8086`, en général le concepteur du système d'exploitation, d'écrire la routine de service correspondante.

Interruptions logicielles.- Ces interruptions sont appelées par le programme.

Interruptions externes.- Les périphériques peuvent déclencher une interruption grâce à deux broches du microprocesseur `i8086` :

- La broche appelée `NMI` (pour *NonMaskable Interrupt*) en position haute déclenche une interruption de type `02h`. Cette interruption n'est pas masquable, c'est-à-dire qu'elle peut intervenir à tout moment (on ne peut pas ne pas en tenir compte).
- La broche appelée `INTR` (pour l'anglais *INTeRrupt*) en position haute permet d'activer n'importe laquelle des 256 interruptions. Ceci se fait en plaçant le numéro de l'interruption désirée sur le bus des données.

Lorsque l'une de ces deux broches passe en position haute, le microprocesseur termine l'instruction qu'il est en train d'exécuter et passe à la routine de service de l'interruption correspondante.

Remarque.- Les interruptions internes n'étaient pas toutes utilisées dans le `i8086` si bien que certains les ont utilisées pour d'autres propos.

10.5 Masquage d'interruptions

Il peut arriver que l'on ait besoin, lors de l'exécution d'une routine, qu'aucune interruption (nécessairement matérielle) ne soit prise en compte, pour ne pas gêner le bon déroulement du programme. Ceci est possible pour les interruptions externes masquables, c'est-à-dire celles déclenchées par la broche `INTR`. On se sert pour cela de l'instruction :

```
CLI
```

(pour l'anglais *Clear Interrupt Flag*), qui a pour effet de positionner à 0 l'indicateur `IF`. Ceci n'a aucun effet sur les interruptions externes non masquables (déclenchées par la broche `NMI`) ou les interruptions logicielles.

Pour permettre à nouveau l'exécution des interruptions masquables, on utilise l'instruction :

```
STI
```

(pour l'anglais *SeT Interrupt flag*), qui a pour effet de positionner à 1 l'indicateur `IF`.