Chapitre 9 LES INTERRUPTIONS

1. DEFINITIONS ET GENERALITES

Comment prendre en compte un événement, comment provoquer une rupture de séquence d'exécution d'un processus dans un délai très court ? Une solution : les interruptions.

1.1 Définitions

Def: Une interruption est un signal déclenché par un événement interne à la machine ou externe, provoquant l'arrêt d'un programme en cours d'exécution à la fin de l'opération courante, au profit d'un programme plus prioritaire appelé programme d'interruption. Ensuite, le programme interrompu reprend son exécution à l'endroit où il avait été interrompu.

Le système d'interruption est le dispositif incorporé au séquenceur qui détecte les signaux d'interruption. Ces signaux arrivent de façon asynchrone, à n'importe quel moment, mais ils ne sont pris en compte qu'à la fin de l'opération en cours.

Mécanisme général : Lorsqu'une interruption survient, le processeur achève l'exécution de l'instruction en cours pour les interruptions externes (cf. plus loin), puis il se produit :

1 Sauvegarde du contexte dans une pile :
- adresse de la prochaine instruction à exécuter dans le programme interrompu,
- contenu des registres qui seront modifiés par le programme d'interruption,
- contenu du mot d'état (registre de drapeaux) rassemblant les indicateurs

(tout cela forme le contexte sauvegardé)

2 Chargement du contexte du programme d'interruption (contexte actif) et passage en mode système (ou superviseur)

3 Exécution du programme d'interruption

4 Retour au programme interrompu en restaurant le contexte (commande Assembleur IRET) et en repassant en mode utilisateur.

1.2 Différents types d'interruptions

Les interruptions ne jouent pas seulement un rôle important dans le domaine logiciel, mais aussi pour l'exploitation de l'électronique.

On distingue donc :

• interruptions internes : protection du système et des processus, appelées par une instruction à l'intérieur d'un programme (overflow, erreur d'adressage, code opération inexistant, problème de parité…) *(hardware internes)*

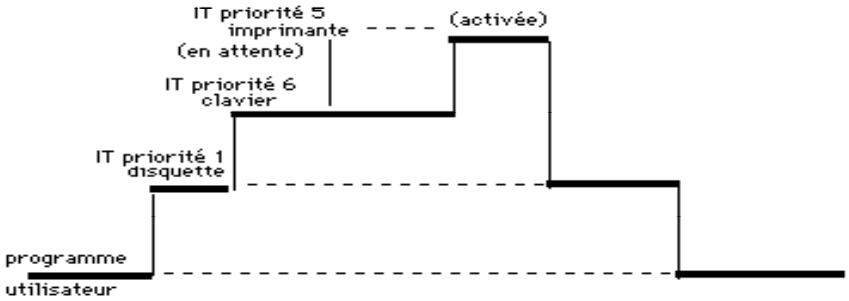
• interruptions logiques :permet à un processus utilisateur de faire un appel au système *(software)*

• interruptions matérielles déclenchées par une unité électronique (lecteur, clavier, canal, contrôleur de périphérique, panne de courant,....) ou par l'horloge *(hardware externes)*

1.3 Les priorités

A chaque interruption, est associée une priorité (système d'interruptions hiérarchisées) qui permet de regrouper les interruptions en classes. Chaque classe est caractérisée par un degré d'urgence d'exécution de son programme d'interruption.

Règle : Une interruption de priorité j est plus prioritaire qu'une interruption de niveau i si j > i.



L'intérêt de ce système est la solution de problèmes tels que :

• arrivée de plusieurs signaux d'interruption pendant l'exécution d'une instruction,

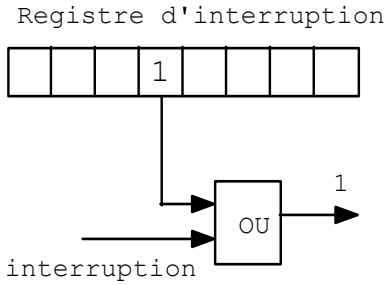
• arrivée d'un signal d'interruption pendant l'exécution du signal de traitement d'une interruption précédente.

On peut utiliser un contrôleur d'interruptions pour regrouper les fils d'interruptions, gérer les priorités entre les interruptions et donner les éléments de calcul d'adresse au processeur.

1.4 Masquage des interruptions

Certaines interruptions présentent tellement d'importance qu'il ne doit pas être possible d'interrompre leur traitement. On **masquera** alors les autres interruptions pour empêcher leur prise en compte. Certaines interruptions sont non-masquables : on les prend obligatoirement en compte. Une interruption masquée n'est pas ignorée : elle est prise en compte dès qu'elle est démasquée.

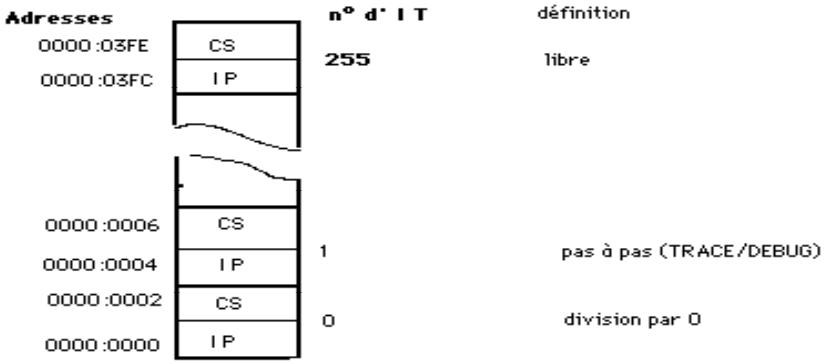
Au contraire, une interruption peut-être **désarmée** : elle sera ignorée. Par défaut, les interruptions sont évidemment armées.



Le bit correspondant à une interruption masquée est à 1 dans le registre d'interruption: sortie de OU à 1, donc masquage. Si le bit est à 0, pas de masquage.

2. INTERRUPTIONS VECTORISEES

Prenons l'exemple du microprocesseur 80386. Il peut comporter 256 interruptions numérotées de 0 à 255. Le lien entre chaque interruption et son programme est réalisé par une table ou vecteur d'interruptions.



Chaque élément contient l'adresse de début d'un programme d'interruption sur 4 octets (offset 2 octets, segment 2 octets). La taille de la table est donc de 4 x 256 = 1 Ko. Elle occupe les adresses 0h à 3FFh en mémoire centrale.

3. LES INTERRUPTIONS MATERIELLES : exemple du Intel 80386 et de ses successeurs

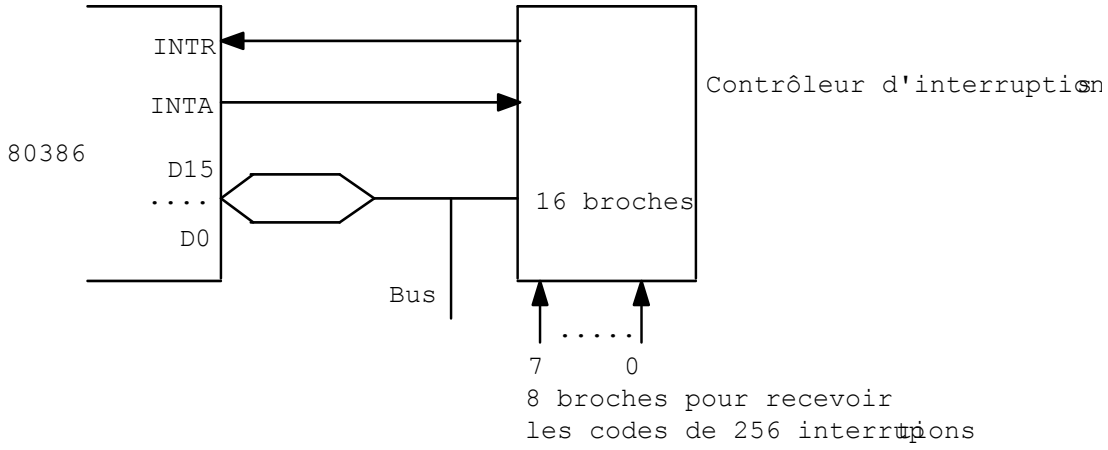
Elles sont déclenchées par une unité électronique. Par exemple, chaque fois qu'on presse ou relève une touche du clavier, l'IT clavier est déclenchée. Le programme d'interruption place le caractère entré dans le buffer à la suite des caractères entrés auparavant. Si le buffer est plein, il déclenche un bip ou une action convenue.

Les interruptions peuvent être masquées (exemple : clavier) grâce à l'instruction Assembleur CLI (CLear Interrupt flag) ou grâce à la fonction TURBO-C ® **disable ()**. Ainsi, après CLI ou disable (), aucun caractère ne peut être entré au clavier. Le flag IF (flag correspondant du registre d'état) est positionné à 0.

De même, l'instruction Assembleur STI (SeT Interrupt flag) ou la fonction TURBO-C ® **enable ()** démasque les interruptions en positionant le flag IF à 1.

La commande Assembleur NMI (No Mask Interrupt) permet de rendre une interruption non masquable. Les interruptions qui ne peuvent être masquables sont toujours exécutables, même après CLI ou disable ().

Dans le contexte Intel, les interruptions sont gérées par un **contrôleur d'interruptions** (circuit intégré 8259 ou circuit inclus dans le microprocesseur lui-même).Ce composant intercepte toutes les demandes d'interruption des périphériques et les communique à l'U.C. selon leurs priorités



INTR : interrupt request : le processeur est informé par le contrôleur de l'arrivée d'une demande d'IT

INTA : interrupt acknowledgement : le processeur a bien reçu par le bus de données le n° d'IT envoyé par le contrôleur. Le mécanisme détaillé de traitement d'une interruption non masquée est le suivant :

- l'U.C. termine l'exécution de l'instruction en cours

- elle lit sur le bus le numéro de l'interruption.

- elle sauvegarde le mot d'état dans la pile et l'adresse de retour CS : IP (adresse de l'instruction suivante)

- elle lit dans la table d'interruption à l'adresse :
[4 * n° d'IT] (mode calculé indirect)
Ainsi, à l'adresse [4*n°d'IT + 2] dans la table, elle trouve la partie segment CS de l'adresse du programme d'interruption et à l'adresse [4* n° d'IT] la partie offset de cette adresse.

- elle met à jour le compteur ordinal avec CS:IP

- elle exécute le programme s'achevant par IRET

Le **ROM BIOS** (Read Only Memory Input/Output System) et le **DOS** représentent le noyau du système d'un micro-ordinateur. Les routines du ROM BIOS permettent essentiellement le traitement des opérations d'entrée/sortie et d'interface avec les périphériques.

Citons notamment les principales interruptions du ROM BIOS :

IT 8 : tic horloge (timer) appelé toutes les 55 ms (18,2 fois/s.). Elle sert par exemple à arrêter le moteur de l'unité de disquette lorsque aucun accès à la disquette n'est 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 ROM BIOS
IT 19h : reset du système
IT 1Ah : gestion de la date et de l'heure

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 d'écran
IT 6 et 7 : inutilisées

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

IT 20h : fin normale d'un programme
IT 21h : appel de sous-routines système
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. La fonction de Turbo-C void **keep** (int retour, in taille) , prototypée dans dos.h, suspend le programme en cours d'exécution, le rend résident. taille désigne sa taille qui peut être déterminée par un appel à MK_FP et code désigne le code retourné à l'environnement par ce programme.

4. ACTIVATION D'INTERRUPTIONS A PARTIR D'UN PROGRAMME C

Fonction **int86** (int num, union REGS * r_in, union REGS * r_out) : génère l'interruption num en fonction des contenus des registres passées en paramètres via r_in . La structure union REGS est définie dans dos.h

Fonction **int86x** (int num, union REGS * r_in, union REGS * r_out, union REGS * r_seg) : utilisée lorsque l'IT nécessite pour son traitement des zônes de mémoire. Pour cela, les registres d'adressage de segments sont passés comme paramètres

Fonction **geninterrupt** (int num) : déclenche l'IT num sans effectuer de copie de registres, mais en initialisant les registres à l'aide de pseudo variables définies dans dos.h : _ah, _al, _ch, etc...

Fonctions **intdos** (union REGS * r_in, union REGS * r_out) et **intdosx** (union REGS * r_in, union REGS * r_out, union REGS r_seg) : permettent d'appeler directement une routine de traitement de l'IT 21h

Fonction **biosequip** : détermine la liste des équipements disponibles (IT 11h)

Fonction **biosmemory** : calcule l'espace mémoire disponible (IT 12h)

Fonction **biosdisk** (int cmd, int face, int piste, int secteur, int n_secteur, void * buffer) : appelle la fonction cmd de l'IT 13h

Fonction **bioscom** (int cmd, char octet, int port) : appelle la routine cmd de l'IT 14h. octet représente l'octet à écrire si cmd = 1, port désigne le n° de la porte

Fonction **bioskey** (int cmd) : appelle la routine cmd de l'IT 16h

Fonction **biosprint** (int cmd, char octet, int port) : appelle la routine cmd de l'IT 18 h. octet représente l'octet à écrire si cmd = 0, port donne le n° de la porte parallèle

Fonction **biostime** (int cmd, long new_time) : appelle la fonction cmd de l'IT 1Ah. new_time contient la nouvelle heure en nombre de battements si cmd = 1

Il existe en Turbo-C le type **interrupt**. Lorsqu'une fonction est déclarée de ce type, TURBO-C va générer du code pour sauver les registres en début d'exécution et pour terminer la routine avec l'instruction IRET signalant le retour d'une interruption. La routine de service doit éviter un appel aux routines système car le DOS n'est pas réentrant.

La fonction **setvect** (int num, void interrupt (*f) ()) insère la fonction f en tant que routine de service de l'IT num. Pour éviter cette opération, il faut éviter d'être interrompu. Il faut donc faire précéder l'appel à setvect d'un appel à disable () qui masquera toutes les IT. Ensuite, un appel à enable () rendra à nouveau possible l'interruption du programme.