Chap. V : Les interruptions

Laurent Poinsot

UMR 7030 - Université Paris 13 - Institut Galilée

Cours "Architecture et Système"

Nous étudions dans ce chapitre les interruptions matérielles (ou externes), c'est-à-dire déclenchées par le matériel (hardware) extérieur au processeur. Nous nous appuyons ici aussi sur l'exemple du PC.

Nous étudions dans ce chapitre les interruptions matérielles (ou externes), c'est-à-dire déclenchées par le matériel (hardware) extérieur au processeur. Nous nous appuyons ici aussi sur l'exemple du PC.

Nous étudions dans ce chapitre les interruptions matérielles (ou externes), c'est-à-dire déclenchées par le matériel (hardware) extérieur au processeur. Nous nous appuyons ici aussi sur l'exemple du PC.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Les interruptions permettent au matériel de communiquer avec le processeur. Les échanges entre le processeur et l'extérieur (les périphériques) que nous avons étudiés jusqu'ici se faisaient toujours à l'initiative du processeur : par exemple, le processeur demande à lire ou à écrire dans une case mémoire. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms). Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Une interruption est signalée au processeur par un signal éléctrique sur une borne spéciale. Lors de la réception de ce signal, le processeur " traite " l'interruption dès la fin de l'instruction qu'il était en train d'exécuter. Le processeur ne peut pas réagir plus vite, imaginez les conséquences d'une instruction abandonnée à la moitié de son exécution...

Une interruption est signalée au processeur par un signal éléctrique sur une borne spéciale. Lors de la réception de ce signal, le processeur " traite " l'interruption dès la fin de l'instruction qu'il était en train d'exécuter. Le processeur ne peut pas réagir plus vite, imaginez les conséquences d'une instruction abandonnée à la moitié de son exécution...

Une interruption est signalée au processeur par un signal éléctrique sur une borne spéciale. Lors de la réception de ce signal, le processeur " traite " l'interruption dès la fin de l'instruction qu'il était en train d'exécuter. Le processeur ne peut pas réagir plus vite, imaginez les conséquences d'une instruction abandonnée à la moitié de son exécution...

Une interruption est signalée au processeur par un signal éléctrique sur une borne spéciale. Lors de la réception de ce signal, le processeur " traite " l'interruption dès la fin de l'instruction qu'il était en train d'exécuter. Le processeur ne peut pas réagir plus vite, imaginez les conséquences d'une instruction abandonnée à la moitié de son exécution...

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : c'est possible uniquement pour certaines interruptions, nommées interruptions masquables. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur " démasque " les interruptions et les prend alors en compte ;
- à exécuter un traitant d'interruption (interrupt handler). Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient. L'adresse de début du traitant est donnée par la table des vecteurs d'interruptions. Lorsque le traitant a effectué son travail, il exécute l'instruction spéciale IRET (pour Interrupt RETurn) qui permet de reprendre l'exécution à l'endroit où elle avait été interrompue.

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Bornes pour les interruptions

Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.

- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Indicateur IF

À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0, alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Contrôleur d'interruption (1/2)

L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.

Contrôleur d'interruption (1/2)

L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.

Contrôleur d'interruption (1/2)

L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.

Contrôleur d'interruption (1/2)

L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.

Contrôleur d'interruption (1/2)

L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Contrôleur d'interruption (2/2)

Le contrôleur est rélié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest à ne pas confondre avec la borne homonyme du processeur INTR). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (1/3)

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

- 1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
- 2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.
- 3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (2/3)

- 4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
- 5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.
- 6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :
  - sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;
  - met l'indicateur IF à 0 pour masquer les interruptions suivantes ;
  - cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

Déroulement d'une interruption externe masquable (3/3)

- 7. La procédure traitant l'interruption se déroule. Pendant ce temps, les interruptions sont masquées (IF=0).
- 8. La procédure se termine par l'instruction IRET qui permet de reprendre le programme qui avait été interrompu.

Déroulement d'une interruption externe masquable (3/3)

- 7. La procédure traitant l'interruption se déroule. Pendant ce temps, les interruptions sont masquées (IF=0).
- 8. La procédure se termine par l'instruction IRET qui permet de reprendre le programme qui avait été interrompu.

Déroulement d'une interruption externe masquable (3/3)

- 7. La procédure traitant l'interruption se déroule. Pendant ce temps, les interruptions sont masquées (IF=0).
- 8. La procédure se termine par l'instruction IRET qui permet de reprendre le programme qui avait été interrompu.

L'horloge d'un PC peut être considéré comme un périphérique d'un type particulier. Il s'agit d'un circuit électronique cadencé par un oscillateur à quartz (comme une montre ordinaire), qui est utilisé pour gérer l'heure et la date, que de nombreux programmes utilisent.

L'horloge d'un PC peut être considéré comme un périphérique d'un type particulier. Il s'agit d'un circuit électronique cadencé par un oscillateur à quartz (comme une montre ordinaire), qui est utilisé pour gérer l'heure et la date, que de nombreux programmes utilisent.

L'horloge d'un PC peut être considéré comme un périphérique d'un type particulier. Il s'agit d'un circuit électronique cadencé par un oscillateur à quartz (comme une montre ordinaire), qui est utilisé pour gérer l'heure et la date, que de nombreux programmes utilisent.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H.

Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

L'horloge envoie une interruption matérielle au processeur toutes 0,055 secondes (soit 18,2 fois par secondes). Le vecteur correspondant (pour le traitant d'interruption) est le numero 08H. Pour gérer l'heure, l'O.S. installe un traitant pour l'interruption 08H. Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse `0040:006C` en mémoire principale. Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change " automatiquement " 18,2 fois par secondes. En langage C, on pourra utiliser la fonction `time()` qui effectue un appel système pour connaître l'heure courante.

En général, les périphériques qui reçoivent des données de l'extérieur mettent en œuvre un mécanisme d'interruption : clavier, liaisons séries (modem, souris...) et parallèles (imprimantes), interfaces réseau, contrôleurs de disques durs et CD-ROMS, etc.

En général, les périphériques qui reçoivent des données de l'extérieur mettent en œuvre un mécanisme d'interruption : clavier, liaisons séries (modem, souris...) et parallèles (imprimantes), interfaces réseau, contrôleurs de disques durs et CD-ROMS, etc.

Étudions ici très schématiquement le cas d'une lecture sur disque dur, afin de comprendre comment l'utilisation d'une interruption permet de construire un système d'exploitation plus efficace.

Étudions ici très schématiquement le cas d'une lecture sur disque dur, afin de comprendre comment l'utilisation d'une interruption permet de construire un système d'exploitation plus efficace.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- `Répéter :`
  1. `envoyer au contrôleur de disque une demande de lecture d'un bloc de données;`
  2. `attendre tant que le disque ne répond pas (scrutation);`
  3. `traiter les données;`
  4. `afficher les résultats.`

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran. Voici l'algorithme général sans utiliser d'interruption :

- Répéter :
  1. envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
  2. attendre tant que le disque ne répond pas (scrutation) ;
  3. traiter les données ;
  4. afficher les résultats.

Cette méthode simple est appelée entrée/sortie par scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

        regarder si le transfert du disque est
                            terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

L'étape 2 est une boucle de scrutation, de la forme :

```
Répéter :

    regarder si le transfert du disque est
                    terminé.

Tant qu'il n'est pas terminé.
```

La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soit transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :

1. Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
2. envoyer au contrôleur de disque une demande de lecture des données ;
3. faire autre chose (un autre calcul ou affichage par exemple).

Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :

1. Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
2. envoyer au contrôleur de disque une demande de lecture des données ;
3. faire autre chose (un autre calcul ou affichage par exemple).

Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :

1. Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
2. envoyer au contrôleur de disque une demande de lecture des données ;
3. faire autre chose (un autre calcul ou affichage par exemple).

Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :

1. Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
2. envoyer au contrôleur de disque une demande de lecture des données ;
3. faire autre chose (un autre calcul ou affichage par exemple).

Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :

1. Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
2. envoyer au contrôleur de disque une demande de lecture des données ;
3. faire autre chose (un autre calcul ou affichage par exemple).

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Dans ce cas, dès que des données arrivent, le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur, qui arrête temporairement le traitement 3 pour s'occuper des données qui arrivent. Lorsque les données sont traitées, le traitement 3 reprend (IRET). Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.