

Les interruptions

VIII.1 Notion d'interruption

Lorsqu'une machine fonctionne, sous le contrôle d'un système d'exploitation, l'unité centrale est en permanence susceptible d'exécuter un programme. Le problème se pose alors de savoir à quel moment l'unité centrale va pouvoir prendre en compte les événements extérieurs à la séquence d'instructions qu'elle exécute : requête d'un périphérique, frappe d'une touche sur un clavier, insertion d'une disquette, fin d'impression, passage d'un objet devant un capteur, système d'alarme, etc.

A chacun de ces événements correspond une tâche à exécuter par l'unité centrale. Cette tâche est codée sous forme d'une procédure du système d'exploitation. Pour pouvoir exécuter cette procédure il faut que se produise une rupture de séquence (fig. 1). Cette rupture doit avoir lieu dans un délai assez court. Pour les problèmes de type "temps réel" un temps maximum de prise en compte et de traitement doit pouvoir être garanti.

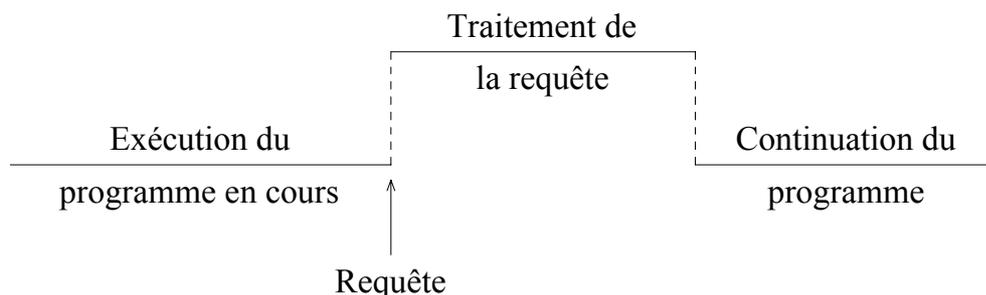


Figure 1

Comment signaler au CPU un événement asynchrone ? Une première approche, dite des drapeaux, consiste à tester périodiquement en séquence chaque unité. Cette technique n'est pas très économique en temps, elle demande une attention constante du CPU et la liste exhaustive des intervenants à tester. Dans une autre approche les machines peuvent être dotées d'un système d'interruptions. Cette technique consiste à pouvoir interrompre "brutalement" l'exécution du programme en cours à la fin de l'instruction courante. Après sauvegarde de son état présent, le contrôle de l'unité centrale est confié à un sous-programme du système dépendant de la nature de l'interruption.

VIII.2 Système d'interruptions hiérarchisées

A la notion d'interruption s'est très rapidement associée la notion de priorité, de manière à ce que certaines requêtes soient servies avant d'autres. Prenons un exemple de la vie courante pour tenter de faire comprendre celle-ci. Imaginons que vous soyez en train d'étudier votre cours (ceci est juste une supposition !). Un camarade arrive pour vous demander un renseignement. Vous terminez la lecture de la phrase en cours, vous mémorisez mentalement où reprendre votre lecture, puis vous vous interrompez pour lui répondre. Vous auriez pu continuer en faisant semblant de n'avoir rien entendu : vous auriez masqué l'interruption. Pendant votre discussion avec votre camarade, un troisième personnage vient vous poser une autre question. S'il s'agit d'un autre camarade vous le laisserez patienter quelques minutes le temps de terminer avec la première demande de renseignement. S'il s'agit de votre professeur vous allez vous interrompre immédiatement (ceci est encore une supposition !) pour pouvoir lui répondre. La demande d'interruption du professeur a un niveau de priorité supérieur, elle peut interrompre l'interruption précédente.

La figure 2 représente un enchaînement de ruptures de séquence provoquées par des interruptions hiérarchisées.

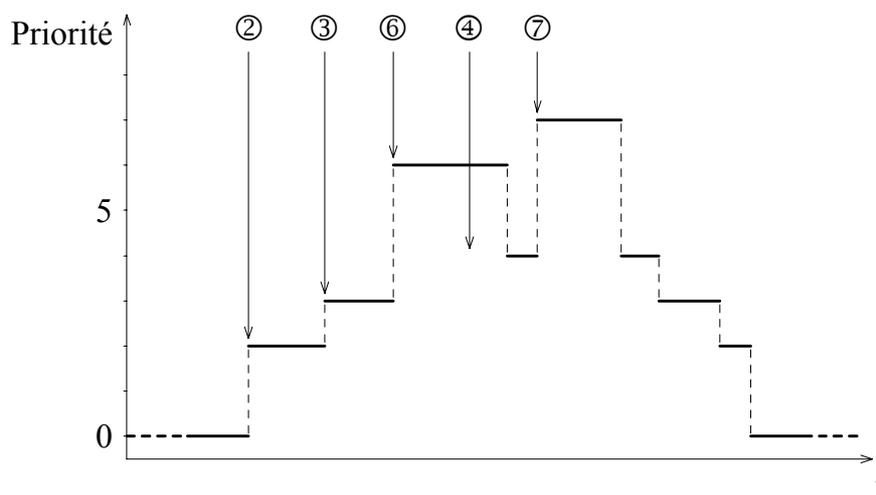


Figure 2

Pendant le déroulement d'un programme intervient une première interruption de niveau 2. Elle est prise en compte, le programme est immédiatement interrompu pour permettre son traitement. Pendant celui-ci un événement extérieur génère une interruption de niveau 3. Le traitement de la première interruption est suspendu pour traiter cette nouvelle interruption. Une nouvelle interruption de niveau 6 se produit pendant l'exécution de la routine associée à l'interruption précédente. Celle-ci est interrompue pour traiter la dernière interruption. Pendant ce traitement une interruption de niveau 4 arrive. Comme celle-ci est de priorité inférieure à celle qui est en cours elle est masquée. Elle n'est prise en compte qu'à la fin du traitement de l'interruption de niveau 6. Elle est ensuite bloquée par une interruption de niveau 7. Son traitement reprend à la fin de celle-ci. L'unité centrale est ensuite rendue au traitement de l'interruption de niveau 3, ensuite à celle de niveau 2 et finalement le programme initialement interrompu peut reprendre son déroulement.

On trouve les systèmes d'interruptions les plus élaborés dans les ordinateurs orientés vers les applications "temps réel" de conduite de processus industriels ou d'acquisition de données. Dans certains cas chaque niveau est découpé en sous-niveaux de priorités différentes. Un bon système doit permettre au programmeur de pouvoir :

- activer/invalider le système d'interruption dans son ensemble;
- armer/désarmer chacune des interruptions individuellement : une interruption désarmée est ignorée;
- masquer/démasquer individuellement chaque interruption : une interruption masquée n'est pas ignorée, elle est mémorisée, mais elle n'est prise en compte que lorsqu'elle est démasquée;
- établir une hiérarchie entre les sources d'interruption avec plusieurs niveaux de priorité, si possible de façon dynamique;
- associer un programme spécifique à chaque interruption.

VIII.3 Les causes d'interruption

Il y a deux grandes catégories d'interruptions : les interruptions externes et les interruptions internes.

Les interruptions externes matérielles sont dues aux périphériques (imprimante, terminal, horloge externe, etc.) ou à des dispositifs extérieurs au système informatique (capteurs, système d'alarme, etc.). Par exemple un périphérique va générer une interruption pour signaler au système qu'il a terminé de traiter la requête précédente et qu'il est prêt pour recevoir une nouvelle commande. Dans le contexte de contrôle de processus, des phénomènes physiques doivent être signalés au système, par exemple le passage d'un objet devant un détecteur. Dans ce cas c'est le dispositif externe au système qui émet un signal qui est reçu par le système comme une interruption.

Le programme courant lui-même peut déclencher une interruption à l'aide d'une instruction spéciale. Cela permet, par exemple, d'activer certains services du système qui nécessitent de bénéficier de privilèges réservés au système. C'est une interruption externe et logicielle.

Les interruptions internes produites par le processeur, sur des erreurs par exemple (division par zéro, dépassement de capacité, erreur d'adressage, accès à une zone de mémoire protégée, etc.). Ces interruptions sont généralement appelées exceptions.

Les interruptions induites par l'horloge permettent la gestion du temps par le système d'exploitation. Par exemple, le système peut programmer l'horloge pour que celle-ci génère une interruption toutes les 10 ms. A chacune de ces interruptions le système peut reprendre la main et exécuter un algorithme de choix de tâches (ordonnanceur) de manière à éviter qu'un programme monopolise le CPU. Ces interruptions peuvent également servir à la gestion du temps dans les programmes et pour la comptabilité.

VIII.4 Mécanisme d'interruption

Décrivons les différentes étapes du traitement d'une interruption.

VIII.4.a Identification de la source d'interruption

L'unité centrale dispose d'une entrée spéciale, généralement appelée IRQ (Interrupt ReQuest), associée à un bit dit bit d'interruption. Avant de passer à l'instruction suivante, le CPU teste l'état de ce bit. S'il est à 1 le CPU est informé d'une demande d'interruption. Pour pouvoir la traiter il doit en déterminer l'origine. Il existe deux méthodes principales pour identifier la source d'une interruption : la scrutation (polling) et l'identification directe. Le choix entre ces deux méthodes est généralement le résultat d'un compromis coût/performance selon en particulier le nombre potentiel de périphériques qui peuvent être connectés. Etudions quelques mécanismes d'identification de la source d'une interruption. De très nombreuses architectures sont possibles.

Dans le premier exemple, basé sur le principe de la scrutation, toutes les lignes d'interruption sont réunies par une porte OU (fig. 3) :

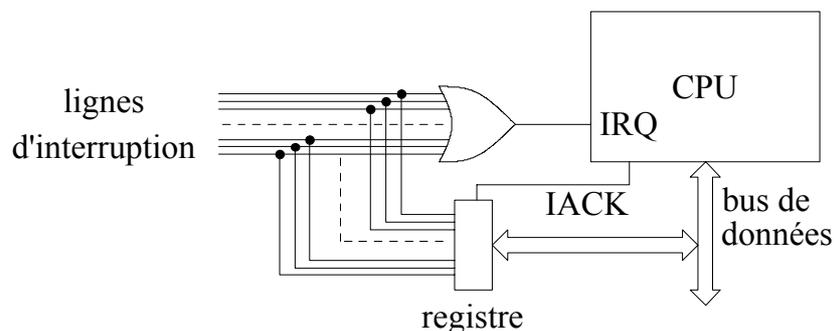


Figure 3

Chacune des lignes correspond à un périphérique. Lorsque le bit d'interruption est trouvé à 1 par le CPU, une routine de service est activée. Celle-ci va tester l'état de la ligne de chacun des périphériques pour identifier le demandeur, puis exécuter la tâche correspondante. Sur l'exemple illustré par la figure précédente, à la réception d'un signal sur l'entrée IRQ le processeur émet un signal IACK qui charge l'état des lignes de demande dans un registre. Le processeur lit ensuite ce registre pour identifier l'origine de la demande. Il peut aussi n'exister qu'une seule ligne d'interruption que chaque périphérique demandeur maintient dans un état haut. L'unité centrale est alors obligée d'interroger chacun des périphériques présents sur le bus pour lui demander s'il requiert un service. La priorité est ici définie par l'ordre dans lequel les lignes sont testées. En général dans ce mode de fonctionnement une interruption n'est elle-même pas interruptible car le mécanisme de scrutation est assez long.

Une autre méthode consiste en un code envoyé par le périphérique (fig. 4). Ce code est utilisé pour exécuter une routine de service spécifique au périphérique. Ce code peut être par exemple un pointeur dans une table contenant le vecteur d'interruption de chaque routine de service. Un vecteur d'interruption contient l'adresse de début de la procédure de traitement de

l'interruption et le registre d'état du processeur nécessaire au début du traitement. Celui-ci, qui peut contenir certains niveaux de privilèges nécessaires au traitement de la requête, remplace le registre d'état courant, après que ce dernier ait été sauvegardé dans la pile avec le reste du contexte du programme interrompu. Le traitement de l'interruption, qui débute à la première instruction spécifique à l'interruption, est donc plus rapide. Très souvent le code envoyé est défini sur le périphérique par un ensemble de petits interrupteurs à positionner au moment de son installation. Dans ce schéma une routine de service peut elle-même être interrompue.

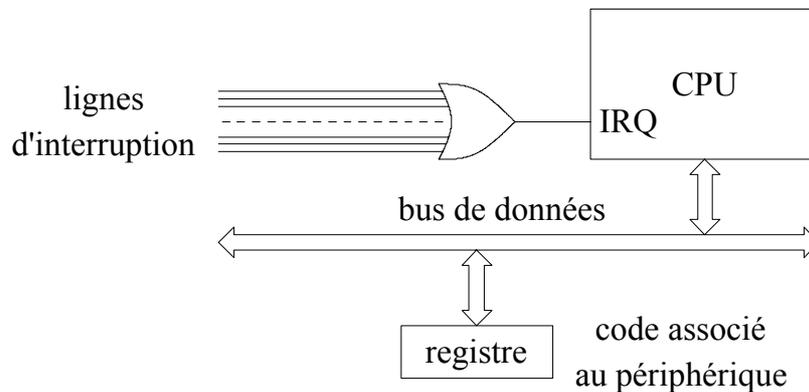


Figure 4

La technique précédente nécessite un mécanisme pour empêcher deux périphériques d'écrire leur code simultanément sur le bus. Il est également possible de chaîner les périphériques (fig. 5). La priorité est alors définie par la position du périphérique sur la ligne d'interruption : daisy-chain ou chaînage de priorité. Par exemple le périphérique P₃ ne peut faire passer sa demande (IRQ) que si P₁ et P₂ n'ont aucune demande en cours. Dans le cas contraire le demandeur en amont empêche la transmission du signal de reconnaissance (IACK) émis par le CPU et qui autorise l'écriture du code sur le bus. La routine de service pour P₃ pourra être interrompue à tout moment par une requête de P₁ ou P₂.

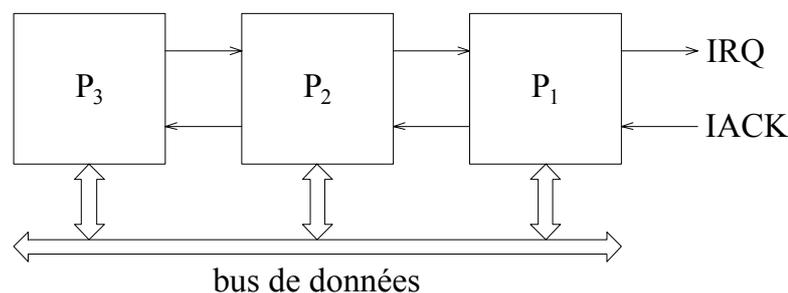


Figure 5

La figure 6 présente le schéma de principe d'une partie d'un contrôleur d'interruption programmable, avec un registre d'entrée qui permet un échantillonnage régulier des signaux d'interruption, ici au nombre de huit. Le second registre contient les interruptions mémorisées qui restent à traiter : OU de lui-même (mémoire) avec le registre d'entrée (nouvelles requêtes). Le processeur peut également programmer un masquage des interruptions en

chargeant un registre de masque. Dans cet exemple, la table des vecteurs est stockée dans une mémoire morte. Le registre de masque et la table des vecteurs sont accessibles par le bus de données. La logique d'effacement des interruptions traitées n'est pas représentée sur ce schéma.

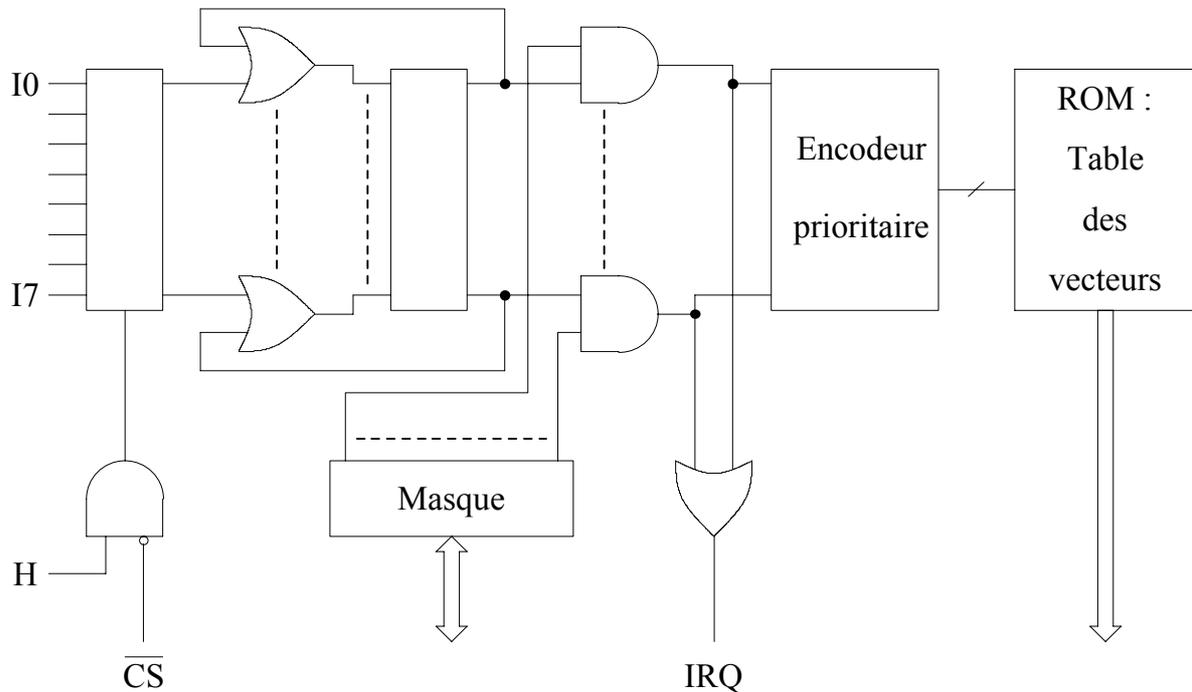


Figure 6

VIII.4.b Actions à entreprendre

Lorsqu'une interruption est détectée il faut que le système réagisse le plus rapidement possible. Pour cela le matériel déclenche un branchement automatique vers une routine système préparée à l'avance. Cette routine se trouve à une adresse fixe qui peut être câblée. L'état de la machine doit être sauvegardé. Le contexte (en particulier, la table des pages, horloge, etc.) du programme interrompu est sauvé, en général dans une pile. Cette sauvegarde du contexte peut être réalisée par la routine de service elle-même (cela permet par exemple : une sauvegarde des registres utilisés uniquement) ou prise en charge automatiquement par l'unité centrale de traitement. En fonction du code identifiant le périphérique responsable de l'interruption l'adresse de la routine de service associée est chargée dans le compteur ordinal, ce qui provoque un branchement automatique. Le registre d'état peut également être modifié pour éventuellement bénéficier de certains privilèges particuliers. Les interruptions de niveau inférieur ou égal doivent être masquées.

Ce mécanisme est rapide car c'est le matériel qui se charge du branchement vers la routine d'interruption, de la sauvegarde du contexte et la modification du compteur ordinal. Il offre une grande souplesse car il est possible d'écrire une grande variété de routines de service. Il peut cependant être pénalisant pour un processeur avec pipe-line et un grand nombre de registres. Dans certaines architectures, en particulier celles orientées vers les

applications temps réel, un processeur supplémentaire peut être associé au processeur principal avec pour tâche de gérer les opérations d'entrées/sorties et de filtrer les interruptions.

VIII.4.c Reprise du programme interrompu.

Le programme en cours est suspendu pendant le temps de traitement de l'interruption. Le contexte doit ensuite être restauré avant la reprise de son exécution. Au retour de la routine de service il faut également démasquer les niveaux masqués à son appel. L'interruption est ainsi totalement transparente. La remise à jour du compteur ordinal provoque le retour au programme à l'instruction suspendue.

VIII.5 Gestion des entrées/sorties

Nous avons vu que les interruptions servent beaucoup à la gestion des entrées/sorties. Nous allons dans ce paragraphe préciser quelques notions supplémentaires concernant la communication avec les périphériques d'entrées/sorties.

Un dispositif d'entrée sortie comprend en général deux parties : un appareil (clavier, écran, disque, imprimante, etc.) et un contrôleur de périphérique. Ce dernier sert d'interface entre l'appareil et le processeur. Il reçoit les requêtes du processeur et les transforme en commandes spécifiques au périphérique, et réciproquement. Ce mécanisme permet une grande souplesse et ne nécessite pas d'instructions spécialisées à chaque type possible de périphérique. L'unité centrale de traitement communique avec un périphérique uniquement en mode lecture ou écriture comme pour la mémoire. Les données proprement dites sont accompagnées de commandes (actions à réaliser) et d'informations (par exemple état du périphérique). Le contrôleur de périphérique est chargé d'interpréter les requêtes ainsi transmises par le processeur et il met également en forme les informations à renvoyer au processeur. Un contrôleur peut être équipé d'un processeur, de registres et d'une mémoire tampon. Il peut être chargé de la gestion des incidents, du contrôle d'erreurs, de la conversion de format, etc. C'est le contrôleur de périphérique qui déclenche l'envoi d'un signal d'interruption.

Par rapport au dialogue on classe souvent les périphériques en deux catégories :

- Les périphériques par caractères, pour lesquels la communication se fait sous la forme d'un flux de caractères par l'intermédiaire de registres.
- Les périphériques par blocs, pour lesquels l'information est accessible par blocs, chaque bloc disposant d'une adresse.

Selon les architectures les adresses des registres ou des blocs peuvent faire partie de la mémoire principale ou correspondre à des zones distinctes à cette mémoire centrale. Ces adresses peuvent être attribuées lors de la mise en place du matériel (par un exemple au moyen de petits interrupteurs ou cavaliers à positionner) ou configurées dynamiquement.

Lorsque les quantités de données à échanger sont importantes, il est pénalisant de charger le processeur de gérer ces échanges. Il est alors préférable de permettre une communication directe entre le périphérique et la mémoire, ou même entre deux

périphériques. Ce mode de communication est appelé DMA : Direct Memory Access. La gestion des échanges sur le bus de données est alors déléguée à un contrôleur DMA.

Sur le plan logiciel on trouve une couche supplémentaire sous le système d'exploitation pour lequel, nous avons dit, le contrôle-commande d'un périphérique peut se limiter à des accès mémoire. Cette couche, appelée pilote ou driver, traduit les commandes générales d'échanges par caractères ou par blocs en commandes spécifiques au périphérique, avec par exemple la gestion des signaux de contrôle de la liaison, des contraintes de temps à respecter, etc. Ces pilotes sont fournis avec le matériel.