INTRODUCTION AU SYSTEME TEMPS REEL

1 Définitions

1.1 Différents types des systèmes

On commence par rappeler les différents types des systèmes informatiques. Ceux-ci sont classés en trois catégories :

- Les systèmes transformationnels : utilisent des données fournies à l'initialisation par l'utilisateur. Le traitement des données et l'obtention des résultats *n'ont aucune contrainte de temps* (calcul scientifiques, gestion de base de données) ;
- les systèmes interactifs : les résultats dépendent des données produites par l'environnement (clavier, fichier, réseaux, souris, etc.). Mais le temps n'intervient pas en tant que tel si ce n'est avec un aspect confort de travail ou qualité de service (outil bureautiques, CAO) ;
- les systèmes temps réel ou réactifs : les résultats sont conditionnés par l'environnement physique réel; mais, dans ce cas, l'aspect « temps » a une place importante sous la forme d'un temps de réaction, d'une échéance à respecter, etc.

On s'intéresse dans ce qui suit aux systèmes temps réel.

1.2 Système temps réel

Un système temps réel est un système dont le comportement dépend, non seulement de l'exactitude des traitements effectués, mais également du temps où les résultats de ces traitements sont produits.

Un système temps réel n'est pas un système « qui va vite / rapide» mais un système qui satisfait des contraintes temporelles (les contraintes de temps dépendent de l'application et de l'environnement alors que la rapidité dépend de la technologie utilisée, celle du processeur par exemple). Les systèmes temps réel peuvent être classés selon les contraintes temporelles en trois types :

Temps réel strict/dur (Hard) : un système dans lequel le non-respect de certaines échéances peut entraîner des conséquences catastrophiques sur le système sous contrôle (ex. : avionique, automobile, transport ferroviaire…) ;

Temps réel Ferme (Firm) : un système temps réel est dite ferme si le non-respect de des échéances ne cause aucun dommage au système, mais la sortie n'a aucune valeur.

Temps réel souple/mou (Soft) : un système dans lequel le dépassement occasionnel des échéances ne met pas le système en difficulté, bien qu'il provoque la dégradation des performances (ex. : jeux vidéo, applications multimédia, téléphonie mobile…).

2 Propriétés

2.1 Prévisibilité

Permet de déterminer à l'avance si un système temps réel va respecter ses contraintes temporelles.

2.2 Déterminisme

Le déterminisme est le but que l'on cherche à atteindre afin de prédire le comportement temporel d'un système temps réel : il s'agit d'enlever toute incertitude sur le comportement des activités individuelles et sur leurs comportements quand elles sont mises ensemble dans le contexte d'exécution du système.

Les sources du non déterminisme :

- Charge de calcul
- Entrées/sorties
- Interruptions
- Fautes et exceptions matérielles ou logicielles

2.3 Fiabilité

La fiabilité est la capacité d'un système qui exécute et maintient ses fonctions dans des environnements normaux, aussi bien que des environnements hostiles ou inattendus.

3 Principales caractéristiques d'un système temps réel

- grande diversité des dispositifs d'entrées/sorties: les données à acquérir qui sont fournies par les capteurs et les données à fournir aux actionneurs sont de types très variés (continu, discret, tout ou rien ou analogique). Il est aussi nécessaire de piloter un bus de terrain pour les communications;
- prise en compte des comportements concurrents: l'ensemble de ces données physiques qui arrivent de l'extérieur et le réseau qui permet de recevoir des messages ne sont pas synchronisés au niveau de leurs évolutions, par conséquent, le système informatique doit être capable d'accepter ces variations simultanées des paramètres ;
- respect des contraintes temporelles: la caractéristique précédente impose de la part du système informatique d'avoir une réactivité suffisante pour prendre en compte tous ces comportements concurrents et en réponse à ceux-ci, de faire une commande en respectant un délai compatible avec la dynamique du système ;
- sûreté de fonctionnement : les systèmes de type contrôle-commande mettent souvent en jeu des applications qui demandent un niveau important de sécurité pour raisons de coût ou de vies humaines. Pour répondre à cette demande, il est nécessaire de mettre en œuvre toutes les réponses de la sûreté de fonctionnement (développements sûrs, tests, prévisibilité, déterminisme, continuité de service, tolérance aux fautes, redondance, etc.).

4 Architecture logicielle des applications temps réel

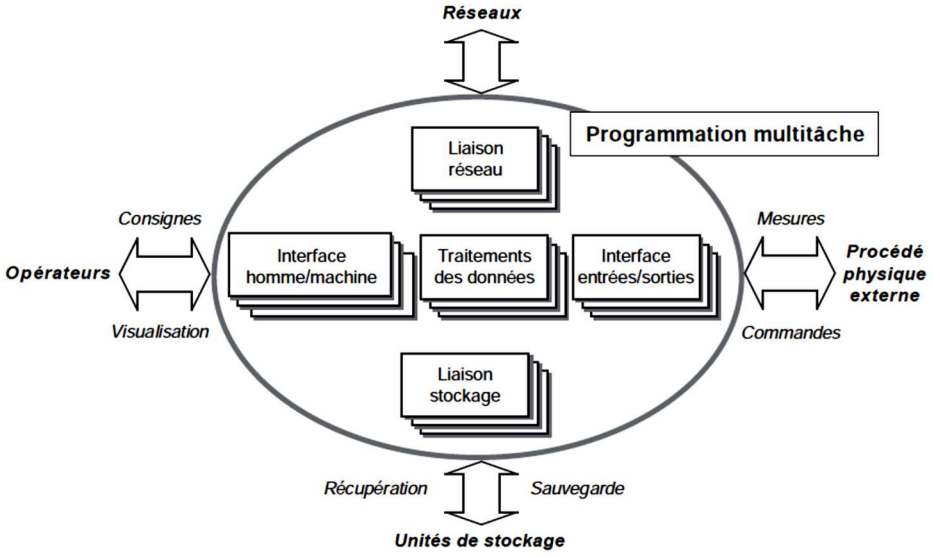
4.1 Architecture multitâche

Un système temps réel typique sera décomposé en tâches, chaque tâche étant responsable de la gestion d'une partie du système. Le comportement concurrent des événements et grandeurs physiques externes amène à décrire l'environnement comme un système fortement parallèle. Cela conduit pour répondre à ce comportement parallèle à adopter une architecture multitâche.

Dans un environnement multitâche, les tâches coopèrent en vue de la réalisation d'une activité commune. Pour une application temps réel, ces tâches peuvent être généralement décomposées selon les groupes des tâches suivantes :

- Tâches d'entrées/sorties : ces tâches permettent d'accéder aux données externes par l'intermédiaire de cartes d'entrées/sorties. Ces tâches peuvent être activées de façon régulière ou par interruption.
- Tâches de traitement : ces tâches constituent le cœur de l'application. Elles intègrent des traitements de signaux (analyse spectrale, corrélation, traitement d'images, etc.) ou des lois de commande (régulation tout ou rien, régulation du premier ordre, régulation PID, etc.).
- Tâches de gestion de l'interface utilisateur : ces tâches permettent de présenter l'état du procédé ou de sa gestion à l'utilisateur. En réponse, l'opérateur peut modifier les consignes données ou changer les commandes. Ces tâches peuvent être très complexes et coûteuses en temps de calcul si l'interface gérée est de taille importante (tableau de bord) ou de type graphique (représentation 2D et 3D).
- Tâches de communications : ces tâches sont destinées à gérer les messages envoyés ou reçus à travers un ou plusieurs réseaux ou bus de terrain. Si ce type de tâches existe, l'application est dite *distribuée* ou *répartie*.
- Tâches de sauvegarde : ces tâches permettent de stocker l'état du système à des instants fixés. Cette sauvegarde peut être utilisée à posteriori pour analyser le fonctionnement de l'application ou lors d'une reprise d'exécution à une étape précédente.

La figure suivante illustre un schéma récapitulatif des groupes des tâches précitées.



5 LES INTERRUPTIONS

5.1 Définition d'une interruption

Une interruption (IT) du programme peut être issue de deux types d'évènements :

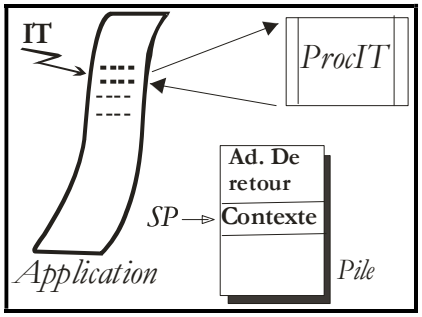
- évènement asynchrone produit par l'environnement extérieur au CPU (interruption matérielle), donc à des instants aléatoires.
- Evènement synchrone produit par l'exécution de certaines instructions (interruption logicielle) donc à des endroits connus dans le programme.

Dans le cas d'un programme temps réel, on s'intéresse particulièrement aux évènements asynchrones matérialisés par des signaux électriques. Une procédure d'E/S peut être invoquée en temps réel par un

évènement asynchrone matériel. Cet évènement est lié à un signal logique au niveau d'une broche spécifique du CPU, appelée entrée d'interruption.

5.2 Rôle des interruptions

Une entrée interruption permet d'interrompre la CPU pour exécuter une tâche (appelée routine d'interruption), considérée comme prioritaire.



Dès l'arrive du signal d'interruption, si celui-ci n'est pas masqué, la CPU charge automatiquement l'adresse de retour dans une zone mémoire appelée PILE ; et charge le compteur de programme PC par l'adresse de routine d'interruption. Une fois l'exécution de la routine d'IT est terminé, la CPU restaure l'adresse de retour à partir de la pile, pour continuer l'exécution de l'application là où elle été interrompue.

Au début de la routine, on devra généralement sauvegarder le contenu de quelques registres. L'ensemble de ces registres s'appelle contexte de la CPU.

L'adresse de la routine d'IT doit se trouver dans une position particulière de la mémoire. En général les adresses des routines d'interruptions sont rassemblées dans une zone particulière de la mémoire, cette zone constitue la table des vecteurs d'interruptions.

5.3 Interruption masquable et non masquable

5.3.1 Interruption non masquable

Une interruption non masquable est prise en compte à son arrivée. Ce type d'interruption est utilisé plutôt dans les fonctionnalités du système : Reset, horloge temps réel, défaillance d'alimentation… Elle est appelée généralement NMI (Non-Maskable interrupts).

5.3.2 Interruption masquable

Ces interruptions sont liées à des fonctionnalités de l'application. La CPU a la possibilité d'honorer ou d'ignorer la demande d'une interruption masquable. En général chaque microprocesseur possède un bit interne, dont sa valeur booléenne détermine l'inhibition (masquage) ou la validation de l'interruption.

- Le microprocesseur 8086 possède un bit I situé dans registre dans le registre d'état.
- Le microcontrôleur 8051 possède un bit EA situé dans le registre IE.
- Les microcontrôleurs PIC possèdent un bit GIE situé dans le registre INTCON.

5.4 Gestion des interruptions

Lors de la gestion de l'interruption, la CPU doit effectuer les procédures suivantes :

- ⇨ Résolution de priorité : la CPU (ou le contrôleur d'interruptions) doit déterminer l'interruption la plus prioritaire au cas où plus qu'une interruption se présente.
- ⇨ Sauvegarde de l'adresse de retour : A cet effet, la CPU doit sauvegarder l'adresse de retour dans une pile (LIFO) gérer par un pointeur de pile avant de se brancher à la routine d'interruption.
- ⇨ Localisation en mémoire de la routine d'interruption : A chaque interruption est associée une adresse permettant de localiser le point d'entrée de la routine d'interruption. Le mécanisme de

création de ces adresses s'appelle vectorisation. La vectorisation est assurée soit par le CPU, soit par l'interface source d'interruption.

5.4.1 Vectorisation par le CPU

C'est la CPU qui fixe les adresses des vecteurs d'interruptions ; ils sont généralement placés en haut ou en bas (selon le type du microprocesseur) de l'espace adressable.

Le schéma de la Figure 1 présente les étapes de déroulement d'une interruption :

- Le périphérique d'E/S génère un signal INT sur la broche d'interruption de la CPU.
- La CPU génère automatiquement l'adresse @vectIT correspondante à cette interruption.
- Il se sert ensuite de cette adresse pour lire l'adresse de la routine d'interruption @procIT à partir de la table des vecteurs d'interruptions.
- La CPU charge l'adresse de la routine d'interruption @procIT dans le registre PC afin de l'exécuter.

Dans le cas des microcontrôleurs PIC et même les microcontrôleurs d'Intel 8051, 8052…, le CPU génère automatiquement l'adresse de la routine d'interruption (sans passer par la table des vecteurs d'interruption, voir Figure 2).

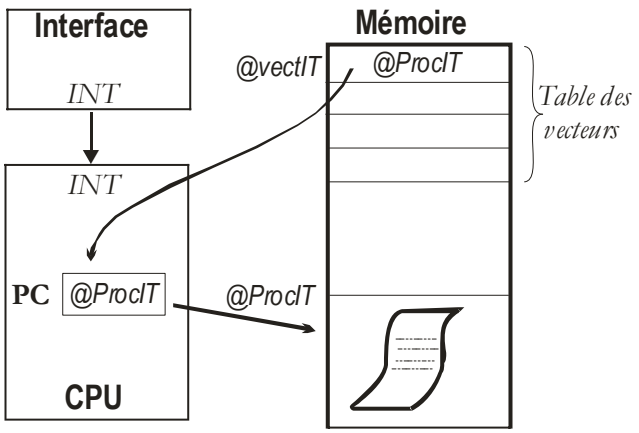


Figure 1

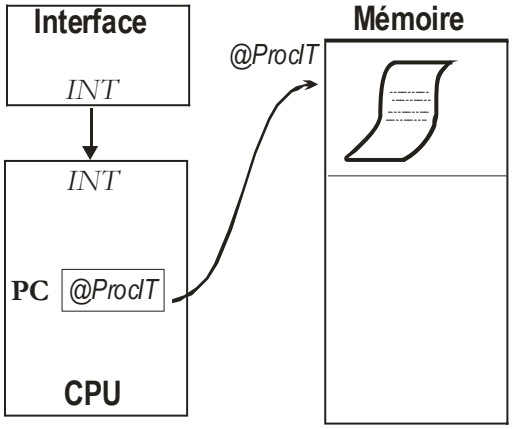


Figure 2

5.4.2 Utilisation d'un numéro de vecteur et d'une table des vecteurs

Dans les microprocesseurs 16-32 bits, après l'accusé de réception du CPU, le contrôleur d'E/S envoi un numéro. A partir de celui-ci la CPU calcule l'adresse physique du vecteur d'interruption (Figure 3) :

- Le périphérique d'E/S génère le signal INT sur la broche d'interruption du CPU.
- La CPU renvoie un accusé de réception IACK.
- A la réception du signal IACK, le périphérique d'E/S envoi à la CPU le numéro du vecteur d'interruption.
- La CPU calcule à partir de ce numéro l'adresse du vecteur d'interruption @vectIT, et, consulte la table des vecteurs d'interruption pour déterminer l'adresse de la routine d'interruption @procIT.
- Après avoir déterminé l'adresse @procIT, la CPU la charge dans le registre PC pour qu'il se branche sur la routine d'interruption.

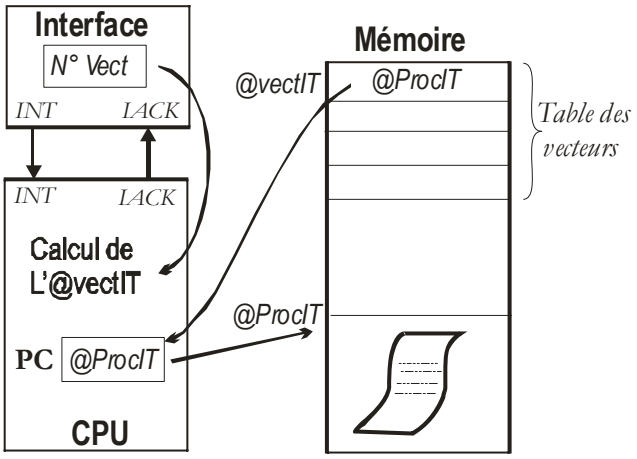


Figure 3

6 Gestion des Entrées/Sorties par Accès direct à la mémoire (DMA)

La gestion est entrées-sorties par interruption est plus efficace que la gestion par scrutation. Cette technique présente des inconvénients lorsque la quantité d'informations à transférer est importante (transfert de données du disque dur, contrôleur d'écran…). Dans ce cas on utilise une troisième technique, appelée Accès direct à la mémoire DMA (pour Direct Memory Access).

L'idée de la DMA est de confier ce transfert à un circuit intégré spécialisé appelé contrôleur DMA (DMAC pour DMA Controller).

Le DMA n'agit pas en parallèle avec la CPU, car il partage avec lui l'ensemble des bus (bus des données, bus des adresses et bus de contrôle). Les bus peuvent être utilisés soit par la CPU, soit par le DMAC mais pas les deux à la fois. Puisque la CPU a le contrôle (primaire) sur les bus, il doit donner la permission au DMAC de les utiliser.

6.1 Modes de transfert des données

Le transfert des données est une opération qui perturbe le séquencement régulier de la CPU, ainsi un contrôleur DMA propose des modes de transfert qui peuvent aller du blocage de la CPU durant tout le transfert de données jusqu'à l'invisibilité du transfert, autrement dit, le DMA effectue des opérations qui sont transparentes pour le CPU et qui ne ralentissent pas son travail.

Transfert par rafale (data burst) : Une fois que le DMA a eu le contrôle du bus système, le transfert de données se poursuit jusqu'à ce que le compteur d'octets transférés atteigne la valeur zéro. La CPU est bloqué pendant le transfert.

Transfert par vol de cycle (cycle stealing) : le CPU est suspendu chaque fois pour la durée d'un cycle de transfert d'une donnée. Ainsi, le CPU est ralenti dans son travail mais pas complètement arrêté durant une longue période.

Mode transparent : les données sont transférées durant des cycles alternés où le CPU n'effectue pas d'accès sur les bus. Ce mode ne ralenti pas le processeur.

7 MEHODES DE GESTION DES ENTREES/SORTIES

De façon théorique, on définit trois méthodes pour répondre à un événement :

- *Par interrogation ou scrutation (Polling en anglais)*

Cette méthode consiste à interroger le registre d'états de chaque périphérique pour déterminer s'il requiert l'attention du processeur.

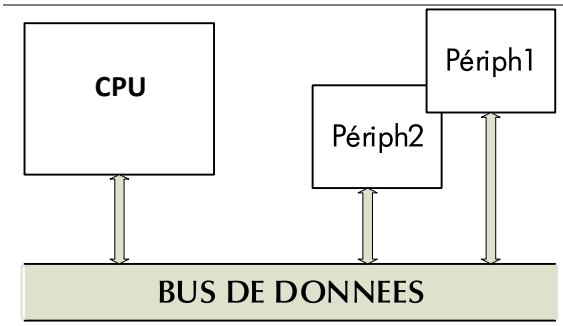
- ✓ *Par interruption à priorité horizontale*

La source de l'interruption est déterminée en interrogeant les registres d'états des périphériques connectés au même niveau de priorité. L'ordre d'interrogation détermine la priorité des périphériques.

- ✓ *Par interruption à priorité verticale*

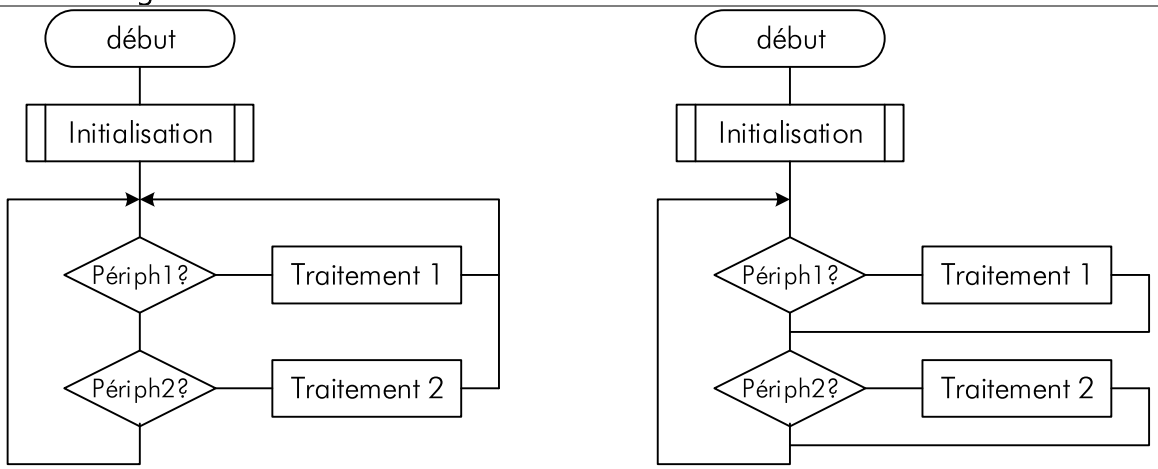
Chaque source d'interruption dispose de sa propre adresse. Une interruption de priorité supérieure peut interrompre une autre interruption.

7.1 Scrutation des entrées/sorties



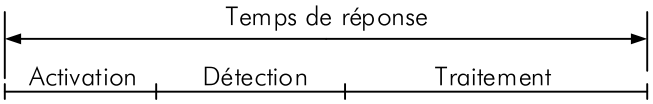
A des points déterminés du programme, la CPU examine régulièrement pour chaque périphérique d'E/S l'état de l'indicateur reflétant l'occurrence d'un évènement extérieur. La lecture des indicateurs se fait d'une manière cyclique. Cet événement est dit synchrone puisque la prise en compte de changement se trouve à des endroits prévisibles (connus) du programme.

Architectures logicielles



Le temps de réponse : c'est la durée de temps qui sépare l'apparition de l'événement et la fin du traitement correspondant. Le temps de réponse se compose de trois éléments :

- Le temps d'activation : ce temps est souvent négligeable (t_s).
- Le temps de détection : il dépend de nombre des périphériques et de l'ordre d'interrogation (t_r).
- Le temps de traitement : dépend du nombre d'instructions que contient le traitement (t_p).



Avantages :

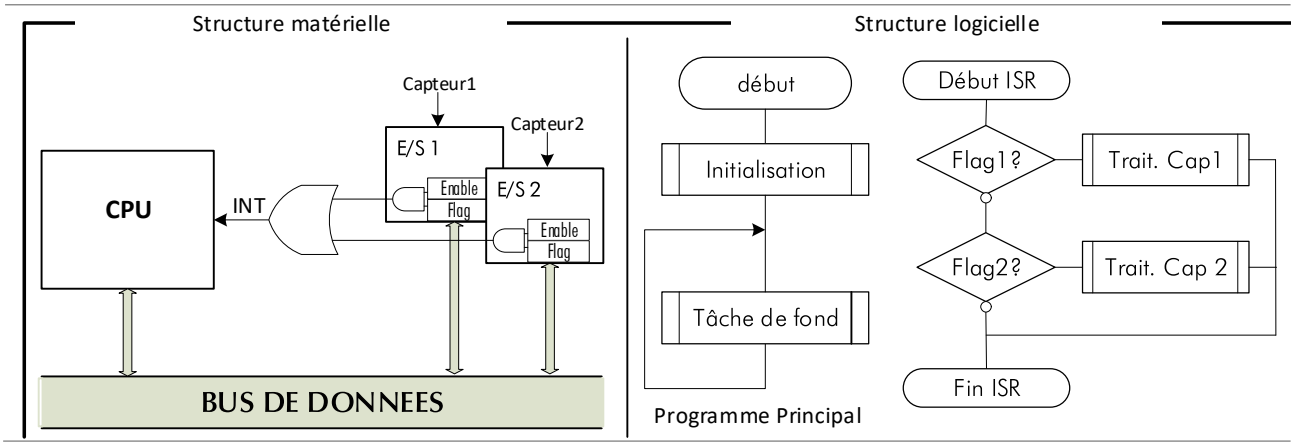
- Temps de réponse plus court lorsqu'il y a un (ou peu) de périphériques (pas la surcharge du changement de contexte)
- Simple à programmer

Inconvénients

- Temps de réponse augmente avec le nombre de périphériques,
- Pas de moyen d'interrompre un traitement.
- Le processeur ne fait pas de travail utile durant l'interrogation "à vide".

7.2 Interruption et interrogation

Plusieurs sources d'interruptions partagent la même entrée d'interruption (OU câblé). Lors de l'arrivé du signal d'interruption, la CPU est prévenu, mais il devra tester les indicateurs (scrutation) pour savoir la source de l'interruption.



TRAITEMENT D'UNE INTERRUPTION

Lorsque la CPU reçoit une demande d'interruption :

1 - Attend la fin de l'instruction en cours.
2 - Désactive/Masque les interruptions.
3 - Copie le Registre d'état (Statut) dans la Pile.
4 - Copie le Compteur de Programme (PC) dans la Pile (empiler l'adresse de retour).
5 - Branchement à la routine d'interruption (La CPU place l'adresse de la routine d'interruption dans le Compteur de Programme).

La routine d'interruption :

8 - Réarme les interruptions.
9 - Sauvegarde le contexte dans la pile (Cs).
10 - Traite l'évènement du périphérique (Ai).
11 - Récupère le contexte (Cr).
12 – Retourne pour poursuivre le programme.

TEMPS DE REPONSE

Le temps de réponse (***Tr***) se compose des éléments suivants :

✓ Latence de l'interruption ***Li*** (dès la détection de l'évènement jusqu'au branchement sur la routine d'interruption).
✓ Sauvegarde du contexte ***Cs***

✓ Traitement de l'interruption $\boldsymbol{Ai}$
- Temps pour détecter le périphérique $\boldsymbol{Tf}$
- Temps pour traiter l'événement $\boldsymbol{Tp}$

✓ Récupération du contexte $\boldsymbol{Cr}$

$$\boldsymbol{Tr = Li + Cs + Ai + Cr}$$

AVANTAGES

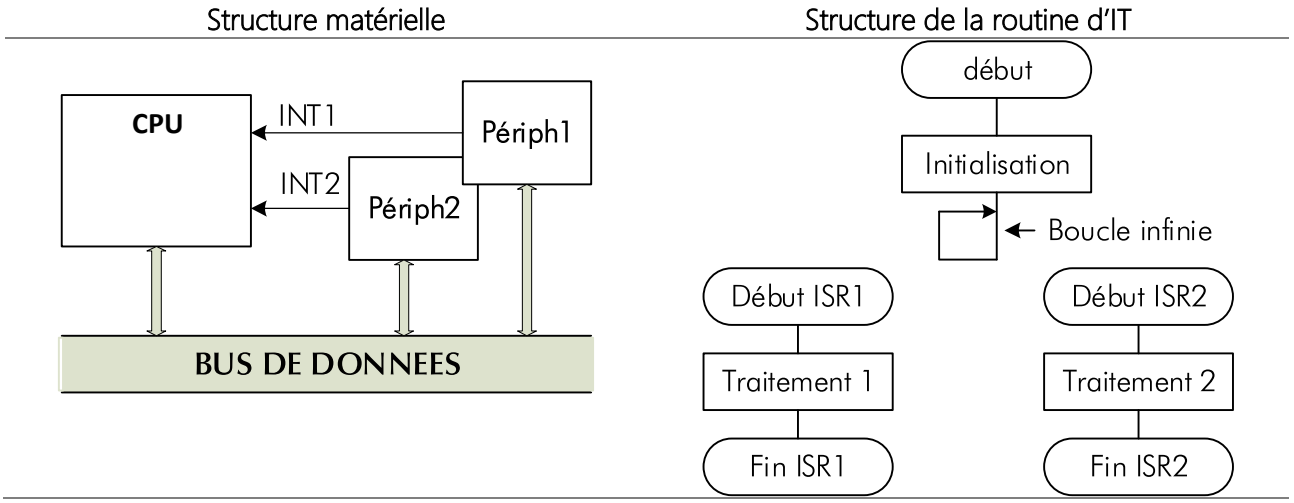
· Plusieurs périphériques utilisent la même ligne d'interruption.
· Possibilité d'interrompre le traitement d'arrière-plan

INCONVENIENTS

· Temps de réponse augmente avec le nombre de périphériques.
· Le processeur ne fait pas de travail utile durant l'interrogation "à vide" (facteur d'utilisation ↓)
· Temps de réponse inclut les délais de latence (Li) et du changement de contexte (Cs + Cr).

7.3 Interruptions vectorisées

Chaque source d'interruption dispose de sa propre entrée. Les adresses des routines d'interruptions sont placées en mémoire dans une zone appelée table des vecteurs d'interruptions.



TEMPS DE REPONSE

$$\boldsymbol{Tr = Li + Cs + Ai + Cr}$$

AVANTAGES

- Temps de réponse généralement plus petit.
- Temps de réponse n'augmente pas avec le nombre de périphériques.
- Plusieurs périphériques peuvent utiliser la même ligne d'interruption.
- Possibilité d'interrompre le traitement d'arrière-plan.
- Possibilité d'interrompre une interruption moins prioritaire.
- Facteur d'utilisation ↑ (pas l'interrogation "à vide")

INCONVENIENTS

- Temps de réponse inclut un délai pour le traitement du vecteur d'interruption.
- Temps de réponse inclut un délai pour le changement de contexte (Cs + Cr).